

**FACULDADE DE ENGENHARIA DA UNIVERSIDADE DO PORTO**

# **Narrative Extraction from Synthetic Clinical Texts in Portuguese**

**Luís Filipe Pires Henriques**



Mestrado em Engenharia e Ciência de Dados

Supervisor: Alípio Mário Guedes Jorge

Supervisor: Nuno Ricardo da Silva Guimarães

October 28, 2024



# **Narrative Extraction from Synthetic Clinical Texts in Portuguese**

**Luís Filipe Pires Henriques**

Mestrado em Engenharia e Ciência de Dados

October 28, 2024



# Abstract

Healthcare professionals, like physicians, handle extensive patient records filled with crucial information for follow-up appointments or procedures. Extracting relevant components from electronic health records (EHRs) can be done using Artificial Intelligence applications based on Machine Learning (ML) techniques. However, developing such solutions is challenging due to privacy protection requirements, bureaucratic procedures, and extensive resources required for creating accurate labels for the specific Natural Language Processing (NLP) task. Those challenges increase the problem of data shortage, where few EHRs annotated are available.

This work tackles this data-shortage problem by using Large Language Models (LLMs) to generate synthetic electronic health records (SEHRs) to augment a training corpus. Generative Pre-trained Transformer (GPT) was the family of models chosen for that task, specifically GPT-3.5 and GPT-4o. The Unified Medical Language System (UMLS) was used to produce a weak annotation tool developed and to augment the prompts with random diseases or conditions. Our proposed approach consists of using GPT-4o as both a generator and annotator of SEHRs. A system/user instruction prompt structure and File Search, which is somewhat similar to a Retrieval-Augmented Generation (RAG) system, were used to create an augmented prompt for GPT-4o. We called this approach the Synthetic Data Generation (SDG) pipeline.

A Portuguese NER dataset annotated by in-domain experts using UMLS, SemClinBr, was used to develop such an approach. We then tested the final SDG pipeline in a European Portuguese NER dataset, PortugueseClinicalNER, with a new Pre-trained Language Model (PLM), MedAI-bertina PT-PT with 900M parameters. We measured NER performance gains against semantic types (STYs) such as Signs or Symptoms, Quantitative Concepts, and Diagnostic Procedures for the first dataset, and Results, Additional Observations, and Characterization for the second.

With the SDG pipeline, results showed NER performance increases relative to the respective baselines with no augmented training data. While outperforming baselines, we were able to reduce significantly the risks of privacy disclosure by using CONCEPT sampling, which consists of adding a random condition to the system/user prompt. We were also able to dramatically close the NER performance gap between data shortage scenarios where one had three times fewer real EHRs for training. Sentence embeddings with t-SNE and cosine similarity showed similarities between real and synthetic sentences. When testing the SDG pipeline on an unbiased NER dataset and with a different PLM, baselines also were outperformed.

**Keywords:** Electronic health records, Synthetic data, Low resources, Natural Language Processing, ChatGPT, GPT-4o, Portuguese, Named Entity Recognition, PLM, UMLS



# Resumo

Os profissionais de saúde, e em particular os médicos, lidam com extensos registros de pacientes repletos de informação crucial para consultas de seguimento ou procedimentos. A extração de componentes relevantes de registros eletrônicos de saúde (EHRs) pode ser realizada utilizando aplicações de Inteligência Artificial baseadas em técnicas de Aprendizagem Automática (ML). No entanto, desenvolver tais soluções é desafiador devido a regulamentos de privacidade, procedimentos burocráticos e recursos extensos necessários para criar etiquetas precisas para a tarefa específica de Processamento de Linguagem Natural (NLP).

Este trabalho aborda o problema da escassez de dados utilizando Grandes Modelos de Linguagem (LLMs) para gerar registros eletrônicos de saúde sintéticos (SEHRs) a fim de aumentar um corpus de treino. O Generative Pre-trained Transformer (GPT) foi a família de modelos escolhida para essa tarefa, especificamente o GPT-3.5 e o GPT-4o. O Unified Medical Language System (UMLS) foi utilizado para produzir uma ferramenta de anotação fraca e para aumentar os prompts com doenças ou condições aleatórias. A nossa abordagem proposta consiste em usar o GPT-4o tanto como gerador quanto como anotador de SEHRs. Uma estrutura de prompt de instrução sistema/usuário e File Search, que é semelhante a um sistema de Geração Aumentada por Recuperação (RAG), foram utilizados para criar um prompt aumentado para o GPT-4o. Chamamos a essa abordagem a pipeline de Geração de Dados Sintéticos (SDG).

Um conjunto de dados de NER português anotado por especialistas da área utilizando o UMLS, o SemClinBr, foi utilizado para desenvolver tal abordagem. Em seguida, testamos a pipeline SDG final num conjunto de dados de NER em português europeu, o PortugueseClinicalNER, com um novo Modelo de Linguagem Pré-treinado (PLM), o MediAlbertina PT-PT, com 900M de parâmetros. Medimos os ganhos de desempenho em NER em relação a tipos semânticos (STYs) como Sinais ou Sintomas, Conceitos Quantitativos e Procedimentos de Diagnóstico para o primeiro conjunto de dados, e Resultados, Observações Adicionais e Caracterização para o segundo.

Com a pipeline SDG, os resultados mostraram aumentos no desempenho de NER em relação aos respetivos baselines com dados de treino não aumentados. Enquanto superávamos esses baselines, conseguimos reduzir significativamente os riscos de divulgação de privacidade ao utilizar a amostragem CONCEPT, que consiste em adicionar uma condição aleatória ao prompt de sistema/usuário. Também conseguimos fechar drasticamente a diferença de desempenho de NER entre cenários de escassez de dados, onde se tinha três vezes menos EHRs reais para treino. Embeddings de frases com t-SNE e similaridade por cosseno mostraram semelhanças entre frases reais e sintéticas. Ao testar a pipeline SDG num conjunto de dados de NER imparcial e com um PLM diferente, os baselines também foram superadas.

**Keywords:** Relatórios de saúde eletrônicos, Textos sintéticos, Baixos recursos, Processamento de Linguagem Natural, ChatGPT, GPT-4o, Português, Reconhecimento de Entidades Nomeada,



# Acknowledgements

It has been a long ride, but like on the road, it has to come to a destination. This ride was not made alone either. Several people were responsible for all the hours, sweat, hard work, motivation, and inspiration for this work to achieve this state. I account you all responsible, and therefore you deserve a special place in this document.

This work was developed in the context of the HealthFromPortugal (HfPT) project in collaboration with INESC TEC. I want to start by giving a special and heartfelt thank you to my advisory team, Alípio Jorge and Nuno Guimarães. You got me selected to my second research scholarship in INESC TEC, and I couldn't be more grateful for that. You were there always questioning my decisions and perspectives, giving constructive feedback, and enlightening me with ways in which I could enrich my dissertation. I also want to thank everyone in the Clinical Narratives team, from people who were there since the beginning to the ones that arrived later.

I also extend my gratitude to the professors in my master's degree. These are high skilled and knowledgeable professionals that spent their time teaching me. Hope I was at least a good enough student to you. Specifically, professors Carlos Soares, and Vera Miguéis. That LCED project was one hell of an uphill battle for sure. A special thanks to the master's director, professor José Luís Moura Borges. To all the friends with which I was closer in my master's degree. We did a lot of projects together. You helped me develop in an environment filled with highly-capable individuals such as yourselves. Hope I could be of marginal help at least. A special thanks to the nerdiest gym rat around, Tomás. I guess we go back a few Earth revolutions already now. Differences aside, I hope I was, and am, a good friend to you.

This was something to which I was not prepared for, and in fact, there is no way I could have ever been. That is exactly the reason why we should always pursue new challenges. It was, and it has been, a very tough and demanding experience. Many times I did not know what to do. Many times I wanted leave it for tomorrow, and I did. Many times I was not sure if I could deliver something that I could be proud of. Many times... I always want to do more and better, and that inner strength pushes me forward. But sometimes, that's not enough. However, I was, and am, fortunate for having someone special that reignites the fire, making me feel strong when I wasn't. If not for me, for her. Thanks, Mom.

Thank you.

Author



*“Until I began to learn to draw,  
I was never much interested in looking at art.”*

Richard P. Feynman



# Contents

|          |                                                       |           |
|----------|-------------------------------------------------------|-----------|
| <b>1</b> | <b>Introduction</b>                                   | <b>1</b>  |
| 1.1      | Context and Motivation . . . . .                      | 1         |
| 1.2      | Objectives . . . . .                                  | 3         |
| 1.3      | Contributions and Innovations . . . . .               | 3         |
| 1.4      | Dissertation structure . . . . .                      | 4         |
| <b>2</b> | <b>Background</b>                                     | <b>5</b>  |
| 2.1      | Outline . . . . .                                     | 5         |
| 2.2      | Synthetic Data Generation . . . . .                   | 6         |
| 2.2.1    | Generative Models . . . . .                           | 7         |
| 2.2.2    | Prompt Engineering . . . . .                          | 15        |
| 2.2.3    | Privacy Disclosure and Realism . . . . .              | 21        |
| 2.3      | Narrative Extraction . . . . .                        | 25        |
| 2.3.1    | Pre-processing and Parsing . . . . .                  | 25        |
| 2.3.2    | Identification and Extraction of Components . . . . . | 36        |
| 2.3.3    | Modeling Techniques . . . . .                         | 44        |
| 2.3.4    | Evaluation . . . . .                                  | 50        |
| 2.4      | Summary . . . . .                                     | 52        |
| <b>3</b> | <b>Literature Review</b>                              | <b>53</b> |
| 3.1      | Outline . . . . .                                     | 53        |
| 3.2      | Synthetic Data Generation . . . . .                   | 54        |
| 3.2.1    | Generative Models . . . . .                           | 54        |
| 3.2.2    | Prompt Engineering . . . . .                          | 56        |
| 3.2.3    | Privacy Disclosure and Realism . . . . .              | 57        |
| 3.3      | Narrative Extraction . . . . .                        | 59        |
| 3.3.1    | Pre-processing and Parsing . . . . .                  | 60        |
| 3.3.2    | Identification and Extraction of Components . . . . . | 61        |
| 3.3.3    | Evaluation . . . . .                                  | 65        |
| 3.4      | Summary . . . . .                                     | 66        |
| <b>4</b> | <b>Work Methods</b>                                   | <b>67</b> |
| 4.1      | Outline . . . . .                                     | 67        |
| 4.2      | SEHRs document generation and annotation . . . . .    | 67        |
| 4.2.1    | GPT-3.5 . . . . .                                     | 68        |
| 4.2.2    | UMLS-based weak annotation . . . . .                  | 69        |
| 4.2.3    | GPT-4o . . . . .                                      | 74        |
| 4.2.4    | Resemblance and privacy disclosure . . . . .          | 77        |

|          |                                                                                                                                            |            |
|----------|--------------------------------------------------------------------------------------------------------------------------------------------|------------|
| 4.3      | Participant extraction . . . . .                                                                                                           | 78         |
| 4.3.1    | NER dataset and STY selection . . . . .                                                                                                    | 78         |
| 4.3.2    | Experiment schemes . . . . .                                                                                                               | 79         |
| 4.3.3    | Pre-processing . . . . .                                                                                                                   | 80         |
| 4.3.4    | Training Pipeline . . . . .                                                                                                                | 83         |
| 4.3.5    | Evaluation . . . . .                                                                                                                       | 87         |
| 4.4      | Summary . . . . .                                                                                                                          | 87         |
| <b>5</b> | <b>Results</b>                                                                                                                             | <b>89</b>  |
| 5.1      | Outline . . . . .                                                                                                                          | 89         |
| 5.2      | Initial approach . . . . .                                                                                                                 | 89         |
| 5.2.1    | SEHRs document generation . . . . .                                                                                                        | 90         |
| 5.2.2    | UMLS-based weak annotations . . . . .                                                                                                      | 94         |
| 5.2.3    | Participants extraction . . . . .                                                                                                          | 95         |
| 5.3      | Comparative Study and Proof of Concept with an LLM as SEHR generator and<br>annotator . . . . .                                            | 98         |
| 5.3.1    | SEHRs document and annotations generation . . . . .                                                                                        | 99         |
| 5.3.2    | Participant extraction . . . . .                                                                                                           | 100        |
| 5.4      | Further SDG adjustments and final results for SemClinBr . . . . .                                                                          | 109        |
| 5.4.1    | SEHRs document and annotation generation adjustments . . . . .                                                                             | 109        |
| 5.4.2    | Final SEHR datasets for SemClinBr augmentation . . . . .                                                                                   | 114        |
| 5.4.3    | Participant extraction . . . . .                                                                                                           | 118        |
| 5.5      | SDG final pipeline with different PLM encoder applied to PortugueseClinicalNER<br>5.5.1 SEHRs document and annotation generation . . . . . | 131        |
| 5.5.2    | Participant extraction . . . . .                                                                                                           | 135        |
| 5.6      | Summary . . . . .                                                                                                                          | 139        |
| <b>6</b> | <b>Conclusions and Future Work</b>                                                                                                         | <b>141</b> |
| 6.1      | Summary of the work developed . . . . .                                                                                                    | 141        |
| 6.2      | Conclusions . . . . .                                                                                                                      | 142        |
| 6.3      | Further Work . . . . .                                                                                                                     | 142        |
| <b>A</b> | <b>Appendix - SEHR examples and test result selection</b>                                                                                  | <b>145</b> |
| A.1      | SEHR examples . . . . .                                                                                                                    | 145        |
| A.1.1    | Example 1 . . . . .                                                                                                                        | 145        |
| A.1.2    | Example 2 . . . . .                                                                                                                        | 148        |
| A.1.3    | Example 3 . . . . .                                                                                                                        | 151        |
| A.2      | Test result selection . . . . .                                                                                                            | 155        |
| A.2.1    | Example 1 . . . . .                                                                                                                        | 155        |
| A.2.2    | Example 2 . . . . .                                                                                                                        | 156        |
| A.2.3    | Example 3 . . . . .                                                                                                                        | 157        |
|          | <b>References</b>                                                                                                                          | <b>159</b> |

# List of Figures

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              |     |
|------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 2.1  | Transformer Network architecture, from Vaswani et al. (2017)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                 | 10  |
| 2.2  | Generative Pre-Training Transformer architecture (left) and input transformations for different downstream NLP task fine-tuning (right), from Radford et al. (2018)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                          | 13  |
| 2.3  | InstructGPT 3-step training diagram, from Ouyang et al. (2022)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 14  |
| 2.4  | Chain-of-Thought summary, from Wei et al. (2022)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             | 18  |
| 2.5  | Self-Consistency relative to FS-CoT prompting, from Wang et al. (2023)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 18  |
| 2.6  | Generated Knowledge Prompting summary, from Liu et al. (2022)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                | 19  |
| 2.7  | A neural-based binary sentence tokenization architecture, from Wicks and Post (2021)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                         | 27  |
| 2.8  | A schematic of the POST task where a token sequence is processed by an encoder together with a classification head (POS tagger) that predict a tag for each input token, from Jurafsky and Martin (2009)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 36  |
| 2.9  | MEMMs (left) vs CRF (right), from Lafferty et al. (2001)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     | 45  |
| 2.10 | BERT pre-training and fine-tuning procedures, from Devlin et al. (2019)                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                      | 46  |
| 4.1  | Diagram of the overall work method followed to generate and annotate SEHRs. CONCEPT is a disease or condition randomly sampled from UMLS to generate SEHRs for it. The SEHR dataset or sample output is the same component used as input in Figure 4.2                                                                                                                                                                                                                                                                                                                                                                                                                                                                       | 68  |
| 4.2  | Diagram of the TRSTR scheme variants. Regardless of what SEHRs are being used, we selected one of the variants of real EHR samples. Orange refers to the TRSTR variants used with SemClinBr. Blue refers to the TRSTR variant used with PortugueseClinicalNER.                                                                                                                                                                                                                                                                                                                                                                                                                                                               | 80  |
| 4.3  | Diagram of the overall procedure followed to achieve the test results for each STY and augmented training corpus. The "Augmented training corpus" input is created considering Figure 4.2. For a given STY, the augmented training corpus and validation/test data are preprocessed. For each restart, a training procedure is started and 2 model instances are retrieved. These can either represent a model status for different epochs or for the same one. Test results are retrieved for each of those model instances. After the three restarts, for a maximum of 6 different model instances, the test performance correspondent to the highest validation strict F1 score of all is selected and used in Chapter 5. | 83  |
| 5.1  | Learning curve of BioBERTpt(all) fine-tuned for the Quantitative Concept STY with only 600 EHRs (baseline).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 96  |
| 5.2  | Learning curve of BioBERTpt(all) fine-tuned for the Diagnostic Procedure STY with only 600 EHRs (baseline).                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  | 97  |
| 5.3  | Dumbbell plot of Entity STY occurrences in the 4 files selected for each prompt                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                              | 111 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                  |     |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.4  | Dumbbell plot of BLEU scores shows a decrease in privacy disclosure by opting by a prompt which comprises a randomness component, CONCEPT . . . . .                                                                                                                                                                                                                                                                                                                              | 112 |
| 5.5  | Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using PCA. The scatterplot on the left corresponds to Prompt GPT-4o adjusted v2 (KL divergence equal to 0.30574), while the one on the right corresponds to Prompt GPT-4o adjusted (KL divergence equal to 0.18830). . . . .                                                                                                               | 113 |
| 5.6  | Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using t-SNE. The scatterplot on the left corresponds to Prompt GPT-4o adjusted v2 (KL divergence equals to 33.49310), while the one on the right corresponds to Prompt GPT-4o adjusted (KL divergence equals to 23.28585). . . . .                                                                                                         | 113 |
| 5.7  | Horizontal barplot with percentage variation of number of entities per STY of gpt-4o-2024-08-06 relative to gpt-4o-2024-05-13. Color encoding helps the perception of which SEHR dataset has more entities per STY. . . . .                                                                                                                                                                                                                                                      | 114 |
| 5.8  | Dumbbell plot of BLEU scores shows very similar values from the BLEU scores for bigrams until 5-grams. These results were achieved considering as references the 600 training EHRs and as candidates the SEHRs generated either by gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . .                                                                                                                                                                                               | 115 |
| 5.9  | Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using PCA. The scatterplot on the left corresponds to gpt-4o-2024-05-13 (KL divergence equal to 0.27300), while the one on the right corresponds to gpt-4o-2024-08-06 (KL divergence equal to 0.09637). . . . .                                                                                                                            | 116 |
| 5.10 | Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using t-SNE. The scatterplot on the left corresponds to gpt-4o-2024-05-13 (KL divergence equal to 15.59174), while the one on the right corresponds to gpt-4o-2024-08-06 (KL divergence equal to 10.43920). . . . .                                                                                                                        | 116 |
| 5.11 | Boxplot of the two samples of averages for the cosine similarity scores between real and synthetic sentences for gpt-4o-2024-05-13 and gpt-4o-2024-08-06. . . .                                                                                                                                                                                                                                                                                                                  | 117 |
| 5.12 | Radial plot of strict F1 scores across all STYs relative to Prompt GPT-4o adjusted, Prompt GPT-4o adjusted v2, and the respective baseline. Such a comparison can be seen on the left radial plot. On the middle radial plot both Prompt GPT-4o adjusted and Prompt GPT-4o adjusted v2 are compared against the 35 SEHRs GPT-4o performance on 100 EHRs. On the right, such a comparison is made against the 35 pSEHRs + GPT-4o. . . . .                                         | 118 |
| 5.13 | Parallel coordinates plot for the differences between strict P and R for Prompt GPT-4o adjusted (left) and Prompt GPT-4o adjusted v2 (right). Prompt GPT-4o adjusted is able to keep strict P and R performance closer than Prompt GPT-4o adjusted v2. . . . .                                                                                                                                                                                                                   | 120 |
| 5.14 | Radial plot of strict F1 scores across all STYs relative to gpt-4o-2024-05-13, gpt-4o-2024-08-06, and the respective baselines with either 600 EHRs (right), 300 EHRs (middle), or 100 EHRs (left). Utility decreases as the number of real EHRs available for training increases. Still, with 600 real EHRs, gpt-4o-2024-08-06 continues to deliver higher performance for strict F1 relative to the baseline, overall. gpt-4o-2024-08-06 achieves the highest utility. . . . . | 121 |

|      |                                                                                                                                                                                                                                                                                                                                                                                                               |     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.15 | Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13. The baseline finds the best balance, but with worse performance. . . . .                                                                                 | 123 |
| 5.16 | Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-05-13 is able to keep strict P and R performance closer than gpt-4o-2024-08-06. The baseline finds the worse balance. . . . .                                                                                                            | 125 |
| 5.17 | Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13 in 2 out of 3 TRSTR schemes. The baseline finds the best balance, but with much worse performance. . . . .                                                | 127 |
| 5.18 | Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13. The baseline finds a good balance, but with a much worse performance on the 100 EHR TRSTR scheme. . . . .                                                | 129 |
| 5.19 | Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13 in 2 out of 3 TRSTR schemes. The baseline finds the best balance, but with much worse performance. . . . .                                                | 131 |
| 5.20 | Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using PCA (left) and t-SNE (right). KL divergence for PCA is equal to 0.21958 and for t-SNE the KL divergence is equal to 11.24558. . . . .                                                                                                                             | 134 |
| 5.21 | Boxplot of the cosine similarity scores between real and synthetic sentences generated by gpt-4o-2024-08-06. The average cosine similarity equal to 0.8252. . . . .                                                                                                                                                                                                                                           | 134 |
| 5.22 | Radial plot of strict F1 scores across all 6 STYs selected relative to the SEHR dataset generated using as LLM gpt-4o-2024-08-06, and the respective baselines. Utility changes depending on which STY is being analyzed. However, overall results show a positive utility with the SEHRs enabling a strict F1 increase across 5 out of 6 STYs. Highest utility achieved for Additional Observations. . . . . | 137 |
| 5.23 | Parallel coordinates plot for the differences between strict P and R for the baseline (left) and gpt-4o-2024-08-06 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than the baseline. . . . .                                                                                                                                                                                    | 137 |



# List of Tables

|      |                                                                                                                                                                                                                                                                                                                                                                                                                                                                                     |     |
|------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 2.1  | Initial state of the current example of BPE . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                               | 31  |
| 2.2  | Second state of the current example of BPE . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                | 32  |
| 2.3  | Third state of the current example of BPE . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                 | 32  |
| 2.4  | Forth state of the current example of BPE . . . . .                                                                                                                                                                                                                                                                                                                                                                                                                                 | 32  |
| 2.5  | Last 5 iterations of the current example of BPE token learner, given the hyperparameter $K = 8$ . . . . .                                                                                                                                                                                                                                                                                                                                                                           | 33  |
| 2.6  | The 17 parts of speech in the Universal Dependencies tagset in English ( <a href="#">de Marnette et al. (2021)</a> ), from <a href="#">Jurafsky and Martin (2009)</a> . . . . .                                                                                                                                                                                                                                                                                                     | 35  |
| 2.7  | Examples of absolute, relative and durational TIMEX, from <a href="#">Jurafsky and Martin (2009)</a> . . . . .                                                                                                                                                                                                                                                                                                                                                                      | 37  |
| 2.8  | Examples of temporal lexical triggers, from <a href="#">Ferro et al. (2005)</a> . . . . .                                                                                                                                                                                                                                                                                                                                                                                           | 38  |
| 2.9  | TIMEX2 tag attributes, from <a href="#">Ferro et al. (2005)</a> . . . . .                                                                                                                                                                                                                                                                                                                                                                                                           | 39  |
| 2.10 | TIMEX3 examples for TIME and DATE tag types, from <a href="#">Pustejovsky et al. (2005)</a> . . . . .                                                                                                                                                                                                                                                                                                                                                                               | 40  |
| 2.11 | Event types and sub-types, from <a href="#">Linguistic Data Consortium (LDC) (2005)</a> . . . . .                                                                                                                                                                                                                                                                                                                                                                                   | 43  |
| 2.12 | Features commonly used in classic feature-based approaches to event extraction, from <a href="#">Jurafsky and Martin (2009)</a> . . . . .                                                                                                                                                                                                                                                                                                                                           | 44  |
| 4.1  | N-grams (up to trigrams) present in the sentence "Relata alucinações somáticas frequentes, onde sente que insetos estão a rasteja sob a sua pele.". The trigram "alucinações somáticas frequentes" has the following sub-n-grams: "alucinações somáticas", "somáticas frequentes", "alucinações", "somáticas", and "frequentes". If the trigram has an exact match in UMLS, for example a symptom, then those other sub-n-grams won't be considered for further annotation. . . . . | 71  |
| 5.1  | Number of occurrences of each STY across the pSEHR dataset consisting of 1,200 SEHRs generated using gpt-3.5-turbo. Up to 5-grams of possible matches were considered. . . . .                                                                                                                                                                                                                                                                                                      | 94  |
| 5.2  | Baseline models, from <a href="#">Schneider et al. (2020)</a> , performance with proper NER classification metrics . . . . .                                                                                                                                                                                                                                                                                                                                                        | 95  |
| 5.3  | Baseline models, using the pre-processing and data splitting procedures developed in this work, performance with proper NER classification metrics . . . . .                                                                                                                                                                                                                                                                                                                        | 96  |
| 5.4  | BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 600 pSEHRs annotated using the UMLS-based weak annotation tool. . . . .                                                                                                                                                                                                                                                                                                                    | 97  |
| 5.5  | BioBERTpt(all) results fine-tuned with an augmented training corpus with 1,200 pSEHRs annotated using the UMLS-based weak annotation tool . . . . .                                                                                                                                                                                                                                                                                                                                 | 98  |
| 5.6  | Number of occurrences of each STY across the four SEHRs samples, either generated by gpt-3.5-turbo or GPT-4o (ChatGPT), as well as either annotated using the UMLS-based weak annotation tool or GPT-4o (ChatGPT). . . . .                                                                                                                                                                                                                                                          | 100 |

|      |                                                                                                                                                                                       |     |
|------|---------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.7  | BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 101 |
| 5.8  | BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 101 |
| 5.9  | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 102 |
| 5.10 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 102 |
| 5.11 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 103 |
| 5.12 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 103 |
| 5.13 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 104 |
| 5.14 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 104 |
| 5.15 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 105 |
| 5.16 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 105 |
| 5.17 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 106 |
| 5.18 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 107 |
| 5.19 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 108 |
| 5.20 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 108 |
| 5.21 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool . .                            | 109 |
| 5.22 | Number of occurrences of each STY across both SEHR datasets generated using gpt-4o-2024-05-13 and gpt-4o-2024-08-06 . . . . .                                                         | 115 |
| 5.23 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2. . . . . | 119 |
| 5.24 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2. . . . . | 119 |
| 5.25 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2. . . . . | 119 |
| 5.26 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2. . . . . | 120 |
| 5.27 | BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2. . . . . | 120 |

|                                                                                                                                                                                |     |
|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.28 BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 122 |
| 5.29 BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 122 |
| 5.30 BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 123 |
| 5.31 BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 124 |
| 5.32 BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 124 |
| 5.33 BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 125 |
| 5.34 BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 126 |
| 5.35 BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 126 |
| 5.36 BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 127 |
| 5.37 BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 128 |
| 5.38 BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 128 |
| 5.39 BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 129 |
| 5.40 BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 130 |
| 5.41 BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 130 |
| 5.42 BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06. . . . . | 130 |
| 5.43 Number of occurrences of each STY for the SEHR dataset generated using gpt-4o-2024-08-06 . . . . .                                                                        | 133 |

|      |                                                                                                                                                                                                                            |     |
|------|----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|-----|
| 5.44 | BLEU score measures obtained considering as references the training, development, and test EHRs . . . . .                                                                                                                  | 133 |
| 5.45 | Baseline results for all STYs of medialbertina_pt-pt_900m fine-tuned with a training corpus of 202 EHRs. In bold are the STYs and their respective baseline performances chosen to be augmented . . . . .                  | 136 |
| 5.46 | medialbertina_pt-pt_900m results fine-tuned for the Results STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed. . . . .          | 138 |
| 5.47 | medialbertina_pt-pt_900m results fine-tuned for the Characterization STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed. . . . . | 138 |
| 5.48 | medialbertina_pt-pt_900m results fine-tuned for the Condition STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed. . . . .        | 138 |
| 5.49 | medialbertina_pt-pt_900m results fine-tuned for the OBS STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed. . . . .              | 139 |
| 5.50 | medialbertina_pt-pt_900m results fine-tuned for the DateTime STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed. . . . .         | 139 |
| 5.51 | medialbertina_pt-pt_900m results fine-tuned for the THER STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed. . . . .             | 139 |
| A.1  | Results of BioBERTpt(all) fine-tuned for the Sign or Symptom STY with an augmented training corpus of 100 real EHRs with 300 SEHRs generated and annotated using gpt-4o-2024-08-06 . . . . .                               | 156 |
| A.2  | Results of BioBERTpt(all) fine-tuned for the Diagnostic Procedure STY with an augmented training corpus of 600 real EHRs with 300 SEHRs generated and annotated using gpt-4o-2024-08-06 . . . . .                          | 157 |
| A.3  | Results of BioBERTpt(all) fine-tuned for the Laboratory or Test Results STY with an augmented training corpus of 100 real EHRs with 300 SEHRs generated and annotated using gpt-4o-2024-05-13 . . . . .                    | 158 |

# Abbreviations

|        |                                                         |
|--------|---------------------------------------------------------|
| ACE    | Automatic Content Extraction                            |
| AE     | Auto-Encoder                                            |
| AI     | Artificial Intelligence                                 |
| BERT   | Bidirectional Encoder Representations from Transformers |
| BiLSTM | Bidirectional Long Short-Term Memory                    |
| BLEU   | Bilingual Evaluation Understudy                         |
| BoW    | Bag-of-Words                                            |
| BPE    | Byte-Pair Encoding                                      |
| BPTT   | Back-Propagation Through Time                           |
| CBOW   | Continuous Bag-Of-Words                                 |
| CGAN   | Conditional Generative Adversarial Network              |
| CHUC   | Coimbra University Hospital Centre                      |
| CNN    | Convolutional Neural Network                            |
| CoNLL  | Conference on Computational Natural Language Learning   |
| CoT    | Chain-of-Thought                                        |
| CRF    | Conditional Random Fields                               |
| CUI    | Concept Unique Identifier                               |
| DD     | Data Driven                                             |
| DL     | Deep Learning                                           |
| DP     | Discourse Parsing                                       |
| EDU    | Elementary Discourse Units                              |
| EHR    | Electronic Health Record                                |
| EL     | Entity Linking                                          |
| EM     | Entity Markers                                          |
| FCNN   | Fully Connected Neural Network                          |
| FN     | False Negative                                          |
| FP     | False Positive                                          |
| FS     | Few-shot                                                |
| FS-CoT | Few-shot Chain-of-Thought                               |
| FSD    | First Story Detection                                   |
| FT     | Fine-tuning                                             |
| GAN    | Generative Adversarial Network                          |
| GDPR   | General Data Protection Regulation                      |
| GIR    | Geographical Information Retrieval                      |
| GKP    | Generated Knowledge Prompting                           |
| GM     | Generative Model                                        |
| GPT    | Generative Pre-training Transformer                     |
| HBD    | Hybrid Driven                                           |
| HTER   | Human-targeted Translation Edit Rate                    |

|        |                                                             |
|--------|-------------------------------------------------------------|
| ICL    | In-Context Learning                                         |
| IDF    | Inverse Document Frequency                                  |
| IR     | Information Retrieval                                       |
| KADL   | Knowledge-Augmented Deep Learning                           |
| KB     | Knowledge Base                                              |
| LCS    | Longest Common Subsequence                                  |
| LDC    | Linguistic Data Consortium                                  |
| LLM    | Large Language Model                                        |
| LM     | Language Model                                              |
| ML     | Machine Learning                                            |
| MLM    | Masked Language Modeling                                    |
| MEMM   | Maximum Entropy Markov model                                |
| METEOR | Metric for Evaluation of Translation with Explicit Ordering |
| MT     | Machine Translation                                         |
| MTB    | Matching The Blanks                                         |
| NED    | Named Entity Disambiguation                                 |
| NER    | Named Entity Recognition                                    |
| NLG    | Natural Language Generation                                 |
| NLP    | Natural Language Processing                                 |
| NLU    | Natural Language Understanding                              |
| NS     | N-shot                                                      |
| NS-CoT | N-shot Chain-of-Thought                                     |
| NP     | Noun Phrase                                                 |
| NSP    | Next Sentence Prediction                                    |
| OIE    | Open Information Extraction                                 |
| PCA    | Principal Components Analysis                               |
| PE     | Positional Embeddings                                       |
| PET    | Pattern-Exploiting Training                                 |
| PLM    | Pre-trained Language Model                                  |
| PPO    | Proximal Policy Optimization                                |
| POS    | Part-Of-Speech                                              |
| POST   | Part-Of-Speech Tagging                                      |
| QA     | Question-Answering                                          |
| RALM   | Retrieval-Augmented Language Model                          |
| RE     | Relation Extraction                                         |
| RL     | Reinforcement Learning                                      |
| RLHF   | Reinforcement Learning from Human Feedback                  |
| RM     | Reward Model                                                |
| ROUGE  | Recall-Oriented Understudy for Gisting Evaluation           |
| RNN    | Recurrent Neural Network                                    |
| RST    | Rhetorical Structure Theory                                 |
| SFT    | Supervised Fine-Tuning                                      |
| SD     | Synthetic Data                                              |
| SDG    | Synthetic Data Generation                                   |
| SEHR   | Synthetic Electronic Health Record                          |
| SJR    | Scimago Journal Ranking                                     |
| SOTA   | State of the Art                                            |
| SRL    | Semantic Role Labeling                                      |
| STY    | Semantic Type                                               |

|        |                                             |
|--------|---------------------------------------------|
| TDT    | Topic Detection and Tracking                |
| TER    | Translation Edit Rate                       |
| TF     | Term Frequency                              |
| TF-IDF | Term Frequency-Inverse Document Frequency   |
| TIMEX  | Temporal Expression                         |
| T-SNE  | T-distributed Stochastic Neighbor Embedding |
| TRSTR  | Train Real Synthetic, Test Real             |
| TRTR   | Train Real, Test Real                       |
| TSTR   | Train Synthetic, Test Real                  |
| UMLS   | Unified Medical Language System             |
| WE     | Word Embeddings                             |
| WSD    | Word Sense Disambiguation                   |
| WLCS   | Weighted Longest Common Subsequence         |



# Chapter 1

## Introduction

Much of the important information in hospitals or clinics is kept as free text in Electronic Health Records (EHR). Natural Language Processing techniques are being used to exploit this information for the development of clinical decision support tools. However, the access to clinical records is often difficult, due to privacy protection or immature medical data systems. This problem is particularly acute in lower resource languages, as it is the case of European Portuguese. This motivates the development of a pipeline for the synthetic generation of EHR.

This dissertation exploits the most recent developments in Natural Language Processing (NLP), more specifically transformer-based Large Language Models (LLMs), to augment an already existing benchmark corpus with synthetic data samples so that model performance is increased on test data. This is a very recent field with still very few developments, especially during the time this dissertation started to be developed.

### 1.1 Context and Motivation

To better understand what problem this dissertation tackles, it is important to first understand what context gives rise to such a problem. To do so, let's start by giving a concrete example that directly showcases the domain in which this work focuses. For such an example, The reader should place himself/herself in the shoes of a physician at a hospital or medical unit. Let's consider, for a moment, that the reader is a physician dealing with a single patient simultaneously.

Patients usually have a historical health or medical record. Let's consider a small patient record with health records that comprise a total of 5,000 words. If the reader has a reading speed of 300 words per minute, which can be considered above average, then it would take the reader about 16 minutes and 30 seconds to read all those 5,000 words. Although it could be said that the physician does not need to read all the words to understand the information written, the truth is that not only do those reports not always have a pre-defined structure, but they also go straight to the point. This means that jumping ahead could mean not understanding the full medical history. For the sake of

argument, if the reader is a very experienced physician, we could say that that time could maybe be reduced by half, or even by more than half, for example, 10 minutes.

Even if it takes the reader 6 minutes and 30 seconds to extract all the relevant information with high effectiveness, this is just for one patient. What if the reader had to follow more than 20 new patients each day, while on top of that, managing other 10 patients who were hospitalized?

This is where the idea of narrative extraction from health records comes from, where the goal is to retrieve the main narrative behind a medical history, such as previous drugs and dosages given to a patient, as well as previous diseases and what drugs were given for each one. However, what if the extraction of relevant elements of the narrative and how they relate to each other, could be done automatically, much faster, and ready to be accessed by the reader in a structured way? Thus, the reader could understand the health record of a patient much faster, given the narrative had been already extracted, and all the reader needed to focus on was the key elements of it.

This dissertation's work was made as part of an INESC TEC research scholarship, specifically part of the HealthFromPortugal<sup>1</sup> (HfPT) project, where our goals consists in the extraction of structured information, more specifically narrative components, from free-form text. We will then summarize them in a dashboard that could be accessed easily by a physician, in Portugal. Given the master's degree to which this dissertation was submitted, and considering the requirements of the main stakeholder, the project's client, such product will be based on several Machine Learning (ML) models applied to NLP, thus being more specifically called Deep Learning (DL) models considering the multi-layered and complex architectures employed by such models to NLP. More specifically, this dissertation's work will focus specifically on the identification and extraction of relevant text annotations, or text spans, in an NLP downstream task called Named Entity Recognition (NER). However, there is a catch to it.

The problem with developing such NLP products is that they are very data-hungry. This means that such models need a considerable amount of text for them to perform well enough so they can be deployed in the real world. This is a problem because privacy regulations and bureaucratic procedures, although important and crucial to be followed, make it very difficult for developers to get access to the necessary amount of text samples necessary to develop well-performing products based on Artificial Intelligence (AI) approaches. This is especially true in the medical domain. Not only that but even if developers had access to a good enough number of medical records, for such records to be considered useful from an AI perspective they would need to be annotated. The annotations are crucial because even with a large sample of medical texts if the labels are not appropriate (for example proper granularity, applicability, and correctness) then the AI-based product won't yield much usability.

This is where this dissertation comes in. Besides the identification and extraction of relevant pieces of text from health reports, the other main component of this dissertation is to develop a way of overcoming the data shortage problem.

---

<sup>1</sup><https://transparencia.gov.pt/pt/fundos-europeus/prr/beneficiarios-projetos/projeto/02/C05-i01.01/2022.PC644937233-00000047/>

## 1.2 Objectives

The main objectives for this dissertation are two, one for each main component of this dissertation. The first consists of dealing with the shortage of data where there is no access to a considerable amount of EHRs with good enough annotations so that model performance can be increased. The second component consists exactly of the narrative extraction part of this work, where proper pre-processing, data mining, and evaluation procedures need to be developed. In our work, narrative extraction will consist of extracting participants of a narrative, Named-Entity Recognition (NER), such as symptoms and test results.

We tackled the first objective using LLMs to generate synthetic samples of electronic health records (SEHRs), together with a weak annotation procedure using an external data source. In this case, we use the Unified Medical Language System (UMLS). For this proof of concept, the LLMs used were the ones available in the ChatGPT API of OpenAI API.

The second objective is to develop a narrative extraction pipeline specifically targeted at a specific NLP task, NER, with proper training, validation, and testing procedures, together with an evaluation module. The work was directed to use transformer-based architectures for token classification in the context of NER.

To define our research questions, the PICOT framework ([Riva et al. \(2012\)](#)) was used. This framework was used considering it was specifically mentioned in a course to prepare students for the dissertation. PICOT is an acronym where each letter refers to a critical component of a research question. PICOT stands for Population, Intervention, Comparison group, Outcome, and Time frame. Considering the context, the two main components of this dissertation, and the objective defined for each one, these were the answers given to each critical component of the research question following the PICOT framework:

- Population:** Portuguese clinical or medical texts (health reports).
- Intervention:** Generation of synthetic health reports using OpenAI LLM's for NER using Transformer-based models.
- Comparison group:** Baseline without synthetic samples.
- Outcome:** Higher performance relative to the baselines.
- Time frame:** Not formally defined.

The answers of the PICOT framework were used to define the flowing research question.

“How can the usage of synthetic health report samples generated using OpenAI's LLMs improve the effectiveness of Transformer-based models for NER from Portuguese health reports relative to their baselines with no synthetic data?”

## 1.3 Contributions and Innovations

Starting with the innovations, which in this case consists of a singular main innovation, it would be the development of a framework for generating synthetic health report samples, or SEHRs,

together with an annotation module, so that developers overcome the data shortage problem and can improve model performance on clinical NER tasks without the necessity of real health reports.

As for the contributions made with this dissertation, these are summarized as follows:

1. A thorough research of both the background and literature up to the first semester of 2024 regarding the two objectives of our dissertation, mainly the first one.
2. A framework to automatically generate and annotate SEHRs in quantity and quality enough to improve the performance of their baseline counterparts with no influence of such synthetic samples.

## 1.4 Dissertation structure

As for the way this dissertation document is organized, it is divided into a total of 5 chapters, excluding this one, Chapter 1. A brief explanation of each chapter's content is presented below:

**Chapter 2, Background:** Provides the theoretical backbone of this dissertation, which is required to understand the several works explored in the Literature Review and the Methodology.

**Chapter 3, Literature Review:** Explores the most recent and relevant developments regarding the two main components of this dissertation, Synthetic Data Generation (SDG) and mainly NER.

**Chapter 4, Work methods:** Takes into consideration the Literature Review, the objectives, and the research question, to develop the methods used in this dissertation.

**Chapter 5, Results:** This chapter showcases the results separated by the two main components of this dissertation while describing several obstacles encountered and considering the work methods.

**Chapter 6, Conclusions and Future Works:** This last chapter summarizes the main conclusions taken from this dissertation's work, for instance reflecting on how this work answered the research question. This chapter also reflects on how further research or applications can be developed following up on the work developed in this dissertation.

## Chapter 2

# Background

This section, as mentioned in Section 1.4, has the task of laying down the theoretical background which will constitute the foundational knowledge required to understand the following chapters, mainly the Literature Review and the Methodology, which are detailed in Chapters 3 and 4, respectively. Although some concepts may be more important than others, understanding this extensive background is important given certain details present in the Literature Review, mainly.

### 2.1 Outline

In this section, the background of this dissertation will be explored. Some of the several topics exposed here will be more important than others, however, they are important to the better understanding of the literature review, in Section 3, but also to several comments that have been done throughout the dissertation.

The background starts with the cornerstones for the generative component of this work, more specifically regarding the details of the generation of synthetic data samples as data augmentation, in Section 2.2. As for the second main component of this dissertation, narrative extraction, several details are presented in Section 2.3.

After an introduction to the topic of synthetic data generation, Section 2.2 starts by exploring the details of several model architectures relevant to the generation of synthetic data, in Section 2.2.1. In Section 2.2.2 the details of the several prompt engineering techniques that exist, from few-shot learning to more advanced prompting techniques such as chain-of-thought. To finish the background of this first main component, Section 2.2.3 explores what the concepts of privacy and realism mean in the context of synthetic data generation, as well as the metrics and schemes used to assess both of them.

As for the background on narrative extraction, it starts by laying down the fundamentals of the pre-processing and parsing techniques, which are used to prepare the text to be used further down a data mining pipeline on NLP, as well as to extract features from it. Such details can be seen in

Section 2.3.1. Next, in Section 2.3.2, the several text component types that can be relevant to be identified and extracted from the prepared text are explained, from time expressions to participants and events. Model architectures that are important to accomplish such tasks and that are crucial for this work are explained in Section 2.3.3. To finish, Section 2.3.4 explores how such tasks can be evaluated.

## 2.2 Synthetic Data Generation

Accessing high-quality medical data sets is essential for creating and developing healthcare digital solutions and opening up new research paths. Yet, acquiring medical data presents a formidable challenge, given the rigorous legal and privacy protocols delineated such as the General Data Protection Regulation (GDPR), detailed in [gdp \(2021\)](#). Such obstacles make it all the more difficult to generate innovative AI solutions using extensive medical data sets [Murtaza et al. \(2023\)](#).

Measures such as masking and anonymization have been applied to safeguard sensitive data [Jaya-balan and Rana \(2018\)](#). However, it has been found that these techniques don't perform with effectiveness, given the lack of truth in the data, and privacy breaches still occur. As a possible solution, researchers are exploring Synthetic Data (SD). This approach can maintain real data's statistical characteristics and structural correlation while significantly reducing the risk of privacy violations and non-conformity with legal regulations [Murtaza et al. \(2023\)](#).

Synthetic Data Generation (SDG) is very complex task, and many recent publications discuss SDG, showing and increasing interest in medical SD with evaluation metrics being developed. However, it is expected a wider adoption of SD for advanced research [Murtaza et al. \(2023\)](#).

Upon closer examination of the SDG process, [Murtaza et al. \(2023\)](#) suggests that it can be broken down into three distinct stages: first, gathering authentic data and information; second, constructing and executing a generative model capable of producing synthetic Electronic Health Records (EHRs); and finally, retrieving the generated records to compile a dataset of artificially generated EHRs.

The second step of the SDG process has three primary methodologies: Knowledge-driven (KD) approaches, data-driven (DD) approaches, and Hybrid (HBD) approaches.

KD approaches utilize publicly available resources such as Knowledge Bases (KBs) or human expertise to generate knowledge. Although KD approaches may not identify hidden patterns in real-world medical data, which could result in losing real-life medical relevance, a knowledgeable curator could mitigate this issue. On the other hand, Generative models (GMs) require curation with rules to ensure that all relevant knowledge is accounted for, with the quality of the GM relying heavily on the skillset of the model curator.

Data-driven (DD) approaches rely solely on real-world data rather than pre-existing knowledge. These methods have gained popularity due to their ability to capture ground-truth patterns and details and their full automation potential. On the other hand, KD approaches can only be partially automated. That being said, DD approaches face a bootstrapping problem - generating synthetic data when there is a lack of actual data. Paradoxically, real data is required to implement

these approaches. Various transformative and simulation approaches use data, but simulation data-driven approaches are the most relevant. These approaches rely on machine learning techniques such as Autoencoders (AE) and Generative Adversarial Networks (GAN) [Murtaza et al. \(2023\)](#).

Hybrid approaches utilize the best theory and data-driven approaches by incorporating knowledge into generative models. Real data is used to train these models, while external advice is given to enhance their convergence [Murtaza et al. \(2023\)](#). This unique combination of KD and data-driven approaches addresses the limitations of each method, such as the need for high-quality synthetic data and the expertise of the model curator. As a result, hybrid approaches offer greater flexibility and can produce more comprehensive synthetic datasets [Murtaza et al. \(2023\)](#).

Researchers consider SD a very viable substitute for real world data as long as it presents two quality attributes [Yale et al. \(2020\)](#): realism and privacy considerations.

Significant progress has been achieved recently in prompt engineering, a methodology that involves structuring input text to enhance the efficacy of Large Language Models (LLMs) in generating output [Chen et al. \(2023\)](#). [Chen et al. \(2023\)](#) overviews various advanced prompt techniques, including Chain-of-Thought (CoT), self-consistency, Generated Knowledge Prompting (GKP), and Retrieval augmentation.

The work developed in this dissertation will give more focus on HBD approaches. Still, emphasis will be given to modeling techniques used in DD approaches.

In Section 2.2.1 the necessary background on the several ML techniques will be given. Section 2.2.2 will explore prompt engineering, while Section 2.2.3 will give the required knowledge on privacy disclosure and realism of SD.

### 2.2.1 Generative Models

Various ML methods, including Recurrent Neural Networks (RNN) with Long Short-Term Memory (LSTM) cells, Autoencoders (AE), Generative Adversarial Networks (GAN), and Generative Pre-trained Transformer 2 (GPT-2), have proven successful in generating Electronic Health Records (EHR) [Murtaza et al. \(2023\)](#).

#### 2.2.1.1 Long Short-Term Memory

In 1997, [Hochreiter and Schmidhuber \(1997\)](#) introduced RNNs with LSTM cells to address the error flow problem encountered during Back-Propagation Through Time (BPTT). This problem, the vanishing and exploding gradient problem, is solved by replacing the RNN cell with an LSTM cell. The LSTM cell includes three gate units that receive as input the sequence at time step  $t$ ,  $x^{(t)}$ , and the hidden state from the previous time step,  $h^{(t-1)}$ . Those gates are the input gate unit ( $i^{(t)}$ ), the forget gate unit ( $f^{(t)}$ ), and the output gate unit ( $o^{(t)}$ ).  $i^{(t)}$  controls which information from the input will affect the memory cell, while  $f^{(t)}$  determines which data should be forgotten from the previous memory cell's state.  $o^{(t)}$  controls the data stored in the current hidden state. Such gate equations correspond to Equation 2.1, 2.2, and 2.3, respectively.

$$Gate_{input} : \quad i^{(t)} = \sigma \left( \mathbf{U}_i x^{(t)} + \mathbf{W}_i h^{(t-1)} + \mathbf{b}_i \right) \quad (2.1)$$

$$Gate_{forget} : \quad f^{(t)} = \sigma \left( \mathbf{U}_f x^{(t)} + \mathbf{W}_f h^{(t-1)} + \mathbf{b}_f \right) \quad (2.2)$$

$$Gate_{output} : \quad o^{(t)} = \sigma \left( \mathbf{U}_o x^{(t)} + \mathbf{W}_o h^{(t-1)} + \mathbf{b}_o \right) \quad (2.3)$$

Besides these three gates, the LSTM also has a cell state candidate,  $g^{(t)}$ , the memory cell,  $c^{(t)}$ , and the hidden states for time step  $t$ ,  $h^{(t)}$ .  $g^{(t)}$  receives the input to develop an initial cell state that will be considered for the final memory cell state, represented in Equation 2.4.  $c^{(t)}$  facilitates information storage and keeps long-term dependencies between steps. Considers the input and forget gate units to compute which parts of the previous memory cell state and current cell state candidate should be forgotten and considered, represented in Equation 2.5.  $h^{(t)}$  comprises a high-dimensional feature space passed to the next step, represented in Equation 2.6

$$MemoryCell_{candidate} : \quad g^{(t)} = \tanh \left( \mathbf{U}_g x^{(t)} + \mathbf{W}_g h^{(t-1)} + \mathbf{b}_g \right) \quad (2.4)$$

$$MemoryCell : \quad c^{(t)} = f^{(t)} \odot c^{(t-1)} + i^{(t)} \odot g^{(t)} \quad (2.5)$$

$$HiddenStates : \quad h^{(t)} = o^{(t)} \odot \tanh \left( c^{(t)} \right) \quad (2.6)$$

### 2.2.1.2 Autoencoder (AE)

As per the research done in Goodfellow et al. (2016), Autoencoders (AEs) are a type of Feed-Forward Neural Networks designed to replicate input to output while preserving some unique features. The network's bottleneck hidden layer reduces the dimension of the input into a latent space, while the reconstruction loss allows the AE to encode as much input data as possible. Therefore, the AE prioritizes relevant input data while avoiding replicating it entirely. The learning process of an AE can be summarized by Equation 2.7:

$$Loss_{AE} : \quad L(x, g(f(x))), \quad (2.7)$$

where  $x$  is the input to be encoded,  $f(x)$  in the encoded input,  $g(f(x))$  is the decoded latent space and  $L$  is the loss function.

More advanced AEs have advanced further by generalizing deterministic into stochastic mappings of the latent variables Goodfellow et al. (2016).

### 2.2.1.3 Generative Adversarial Network (GAN)

The concept of GANs was first introduced in [Goodfellow et al. \(2020\)](#), which presented a novel approach to generating data. Rather than minimizing the distance to a specific input or distribution, GANs use a generator,  $G$ , to produce fake data and a discriminator,  $D$ , to distinguish between real,  $x^{(i)}$ , and synthetic data,  $G(z^{(i)})$ .

$G$  takes input noise,  $z^{(i)}$ , and produces a fake data instance, which the discriminator then evaluates alongside the real training data.  $D$ , outputs a probability score for each instance, indicating the likelihood of it being real or synthetic. Both the generator and discriminator loss functions,  $\mathcal{L}_G$  and  $\mathcal{L}_D$ , respectively, use this score to improve the generator's output until the discriminator cannot distinguish between real and fake data. Equations 2.8 and 2.9 show the loss functions for the discriminator and generator, respectively.

$$\begin{aligned} \text{Loss}_D: \quad \mathcal{L}_D = & -\frac{1}{m_{\text{real}}} \sum_{i=1}^{m_{\text{real}}} y_{\text{real}}^{(i)} \cdot \log \left( D \left( x^{(i)} \right) \right) \\ & -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \left( 1 - y_{\text{gen}}^{(i)} \right) \cdot \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right), \end{aligned} \quad (2.8)$$

where the first cross-entropy component of  $\mathcal{L}_D$  is related to measuring the loss related to real data while the second cross-entropy component measures the loss of fake data.

$$\text{Loss}_G: \quad \mathcal{L}_G = -\mathcal{L}_D = \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log \left( 1 - D \left( G \left( z^{(i)} \right) \right) \right), \quad (2.9)$$

where here the objective of  $G$  is the opposite of  $D$ .

After the introduction of GANs, [Mirza and Osindero \(2014\)](#) developed a conditional version known as CGAN. In addition to  $z^{(i)}$  and  $x^{(i)}$  provided to the  $G$  and  $D$ , respectively, CGAN also incorporates extra information that is concatenated to both inputs, known as a condition,  $c^{(i)}$ .  $c^{(i)}$  can include class labels or data from different modalities, such as text. As such, this generative model can learn to prioritize which features are more relevant to generate. Equations 2.10 and 2.11 show the discriminator and generator loss functions conditioned by  $c^{(i)}$ , respectively.

$$\begin{aligned} \text{Loss}_D: \quad \mathcal{L}_D = & -\frac{1}{m_{\text{real}}} \sum_{i=1}^{m_{\text{real}}} y_{\text{real}}^{(i)} \cdot \log \left( D \left( x^{(i)} \mid c^{(i)} \right) \right) \\ & -\frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \left( 1 - y_{\text{gen}}^{(i)} \right) \cdot \log \left( 1 - D \left( G \left( z^{(i)} \mid c^{(i)} \right) \right) \right), \end{aligned} \quad (2.10)$$

$$\text{Loss}_G: \quad \mathcal{L}_G = -\mathcal{L}_D = \frac{1}{m_{\text{gen}}} \sum_{i=1}^{m_{\text{gen}}} \log \left( 1 - D \left( G \left( z^{(i)} \mid c^{(i)} \right) \right) \right), \quad (2.11)$$

### 2.2.1.4 Transformer Network

In 2017, the Transformer Network was introduced in [Vaswani et al. \(2017\)](#), becoming a significant milestone. Similar to other sequence-to-sequence models, the Transformer network comprises an encoder with multiple encoder layers that map an input sequence of symbols or tokens,  $u = (u_1, \dots, u_n)$ , to a continuous dense vector representation known as embeddings,  $z = (z_1, \dots, z_n)$ , with dimension  $d_{model}$ . Besides the encoder module, the Transformer is also comprised of a decoder with several decoder layers that compute the predicted tokens of the model's output sequence,  $y = (y_1, \dots, y_n)$ . Such architecture is depicted in Figure 2.1.

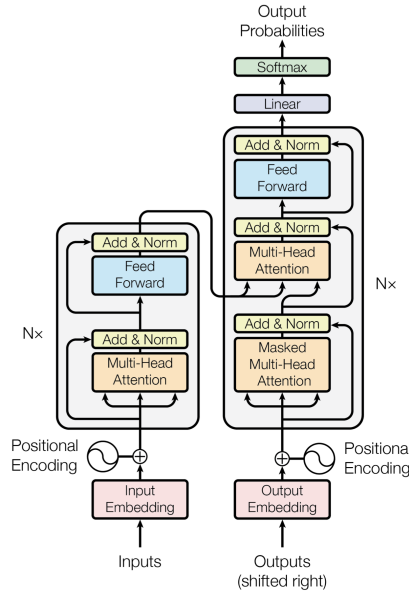


Figure 2.1: Transformer Network architecture, from [Vaswani et al. \(2017\)](#)

The encoder layer has two sub-layers: a self-attention layer and a position-wise Fully-Connected Neural Network (FCNN), shown in Equation 2.12.

$$FCNN_{position-wise} : FCNN(x) = ReLU(xW_1 + b_1)W_2 + b_2, \quad (2.12)$$

Such FCNN has a total of one hidden layer which uses the weight matrix  $W_1$  to transform the input features  $x$  into new features, which then are scaled using the  $ReLU$  activation function. The equation for the  $ReLU$  function can be seen in Equation 2.13.

$$ReLU(x) = \max(0, x) \quad (2.13)$$

Both encoder sub-layers have residual connections that are normalized with layer normalization [Ba et al. \(2016\)](#). Residual connections mean the outputs of a sub-layer are augmented by the

sub-layer's respective inputs via concatenation. The decoder layer has three sub-layers: a masked self-attention layer, a cross-attention layer, and a FCNN like in Equation 2.12. Combined with offset output embeddings, the masked self-attention layer allows for an autoregressive behavior by limiting the encoder to predict the next symbol while considering the previously predicted ones.

The Transformer model employs multi-head attention instead of using scaled dot-product attention once. These receive as input keys,  $K$ , and queries,  $Q$ , with dimension  $d_k$ , and values,  $V$ , with dimension  $d_v$ . Equation 2.14 showcases the scaled dot-product attention layer, while Equation 2.15 the multi-head attention layer and Equation 2.16 the computation of a given head  $i$ , out of  $h$  heads, using Equation 2.14, where  $K$ ,  $Q$ , and  $V$  are parameterized by weight matrices  $W_i^K$ ,  $W_i^Q$ , and  $W_i^V$ , respectively.  $W^O$  parametrizes the final output of Equation 2.15.

$$Attention_{scaled\dot{product}} : \quad Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \quad (2.14)$$

$$Attention_{Multi-head} : \quad MultiHead(Q, K, V) = Concat(head_1, \dots, head_h)W^O, \quad (2.15)$$

$$Attention_{head} : \quad head_i = Attention\left(QW_i^Q, KW_i^K, VW_i^V\right), \quad (2.16)$$

where  $W_i^Q \in R^{d_{model} \times d_k}$ ,  $W_i^K \in R^{d_{model} \times d_k}$ ,  $W_i^V \in R^{d_{model} \times d_v}$ , and  $W^O \in R^{hd_v \times d_{model}}$ .

This attention mechanism allows the model to focus on different input subspaces for different positions, preventing the resulting matrix elements from reaching saturated regions of the softmax. The output from each attention head is concatenated and passed through an affine function without a bias term. Self-attention is used on the encoder or decoder and employs the same input for queries, keys, and values. Masked self-attention is a particular case used on decoder layers, which masks symbols from future positions to condition predictions on previous symbols. Cross-attention uses queries from the previous decoder layer, while the keys and values come from the encoder's output. Such attention mechanism enables the Transformer Network to process entire sequences in parallel, while with recurrency such process is sequential.

The Transformer model utilizes positional encodings (PE), which dimensions  $d_{model}$  and are added to the sequence embeddings. This enables the model to consider the order of the sequence, as it does not utilize recurrency or convolutions but only attention. The PE can be either sinusoidal or learned embeddings. Sinusoidal PE's are shown in Equations 2.17 and 2.18.

$$PE_{even} : \quad PE_{(pos, 2i)} = \sin\left(pos/10000^{2i/d_{model}}\right), \quad (2.17)$$

$$PE_{odd} : \quad PE_{(pos, 2i+1)} = \cos\left(pos/10000^{2i/d_{model}}\right), \quad (2.18)$$

where  $pos$  is the relative position of a given symbol in the input sequence, and  $i$  is the dimension, meaning that each positional encoding has its own sinusoid.

While multi-head attention runs in parallel, the encoder and decoder run sequentially. This is because subsequent encoder and decoder layers rely on the previous ones' outputs as input.

**Training details** As for the training details, the input texts were tokenized using BPE, which is explained in detail in Section 2.3.1.3 in the bottom-up approaches. Two different Machine Translation (MT) datasets were used, resulting in vocabularies with 37,000 and 32,000 tokens.

As the chosen algorithm to optimize the respective loss functions for the downstream NLP tasks, a very well-known gradient descent for stochastic optimization was used, called Adam, developed in Kingma and Ba (2015). Adam with  $\beta_1 = 0.9$ ,  $\beta_2 = 0.98$  and  $\epsilon = 10^9$  were used. A relevant explanation of optimizers and the most relevant for this work is written in Section 2.3.3.4. During training, the learning rate changed so that in the beginning it started increasing linearly from zero until a certain maximum learning rate. Such a stage is called the warmup phase, which occurs for a given number of gradient descent steps. Then, it started reducing proportionally to the inverse square root concerning the optimization step relative to the total number of steps. The learning rate scheduler function can be seen in Equation 2.19.

$$lr_{scheduler} : \quad lrate = d_{model}^{-0.5} \cdot \min(step\_num^{-0.5}, step\_num \cdot warmup\_steps^{-1.5}) \quad (2.19)$$

One specific technique used as a regularization technique to prevent overfitting is called Dropout, introduced in Srivastava et al. (2014). Dropout consists of a procedure that, during training, randomly disables neurons or perceptrons on every layer of a FCNN with a given probability  $p$ , which is called the dropout rate.

### 2.2.1.5 Generative Pre-Training Transformer (GPT)

The GPT model, developed by Radford et al. (2018), utilizes a transformer decoder with twelve layers to build its Language Model (LM). As GPT lacks an encoder, its decoder layers only consist of two sub-layers rather than the three on the original Transformer decoder layer. This is because the multi-head cross-attention layer, which uses the encoder's output as keys and values, is removed. Each GPT decoder layer is instead comprised of a masked multi-head self-attention layer and a position-wise FCNN, both with residual connections. Consider an unsupervised corpus of tokens  $\mathcal{U} = \{u_1, \dots, u_n\}$  and a supervised corpus  $\mathcal{C}$ . GPT's architecture can be seen in Figure 2.2, as well as in Equations 2.20, 2.21, and 2.22.

$$Input_{embeddings} : \quad h_0(\mathcal{U}) = UW_e + W_p, \quad (2.20)$$

$$Transformer_{Decoder} : \quad h_l = transformer_{block}(h_{l-1}) \forall i \in [1, n], \quad (2.21)$$

$$Output : \quad P(u) = softmax(h_n W_e^T), \quad (2.22)$$

where  $U = (u_{-k}, \dots, u_{-1})$  is the context vector of tokens,  $n$  is the number of decoder layers,  $W_e$  is the token embedding matrix, and  $W_p$  is the position embedding matrix.

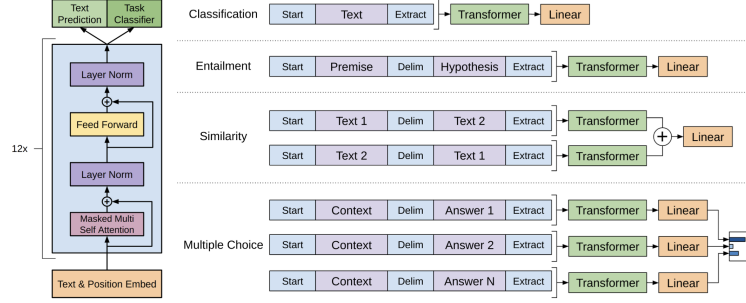


Figure 2.2: Generative Pre-Training Transformer architecture (left) and input transformations for different downstream NLP task fine-tuning (right), from Radford et al. (2018)

To pre-train this LM in an unsupervised manner, a log-likelihood loss function was maximized to predict the next sequence symbol within a context window accurately, shown in Equation 2.24

$$Loss_{unsupervised} : L_1(\mathcal{U}) = \sum_i \log P(u_i | u_{i-k}, \dots, u_{i-1}; \Theta), \quad (2.23)$$

where  $k$  is the context window size and  $P$  is modeled with parameters  $\Theta$ .

Subsequently, GPT was fine-tuned in a supervised environment to accomplish various NLP tasks. Alongside its primary objective of correctly predicting targets, the language modeling loss was also considered a secondary objective, weighted by a lambda scalar,  $\lambda$ . Equation 2.24 represents the fine-tuning objective to be maximized, while Equation 2.25 is the fine-tuning loss function of GPT.

$$Objective_{fine-tuning} : L_2(\mathcal{C}) = \sum_{(x,y)} \log P(y | x^1, \dots, x^m) \quad (2.24)$$

$$Loss_{supervised} : L_3(\mathcal{C}) = L_2(\mathcal{C}) + \lambda * L_1(\mathcal{C}) \quad (2.25)$$

### 2.2.1.6 Generative Pre-Training Transformer 2 (GPT-2)

The creators of GPT-2, as cited in Radford et al. (2019), made some notable changes to the original GPT model's architecture. In the decoder layers, layer normalization was moved to before each sub-layer and residual connection, and an extra layer normalization was added after the final multi-head self-attention layer. GPT-2 also boasts 48 decoder layers and an output dimension of 1600, as opposed to GPT's 12 decoder layers and output dimension of 512. GPT-2 contains 1,542 M parameters and an extended Byte-Pair Encoding (BPE) vocabulary of 50,257 tokens. Additionally, the context window size has increased to 1024 tokens, and the mini-batch size has increased to 512 sequences compared to the original GPT model. Like with GPT, the approach is to compute

the token distribution probability given the context window, as shown in Equation 2.26:

$$\text{DistributionProbability: } p(x) = \prod_{i=1}^n p(s_n | s_1, \dots, s_{n-1}) \quad (2.26)$$

### 2.2.1.7 Generative Pre-Training Transformer 3 (GPT-3)

Introduced in [Brown et al. \(2020\)](#), GPT-3 maintained a similar architecture to GPT-2, some hyper-parameters were adjusted to increase the model size. The number of decoder layers doubled to 96, with 96 attention heads each, and the output dimensions increased to 12,288. The context window size expanded to 2,048 tokens, occasionally encompassing full documents, and the mini-batch size increased to 3.2M sequences. GPT-3 was tested using a Few-Shot (FS) prompting strategy to compare its performance with several SOTA results in translation and question answering. The study tested 8 GPT-3 instances ranging from 125M parameters to 175B parameters, with the latter being the official GPT-3 model.

### 2.2.1.8 Generative Pre-Training Transformer 3.5 and 4 (ChatGPT)

ChatGPT is a sibling of a GPT model developed by [Ouyang et al. \(2022\)](#) called InstructGPT, where Reinforcement Learning from Human Feedback (RLHF) [Christiano et al. \(2017\)](#) together with Proximal Policy Optimization (PPO) [Schulman et al. \(2017\)](#) algorithm were utilized to optimize the policy were used. According to [Ouyang et al. \(2022\)](#), recent LMs such as GPT-3 struggle when following user instructions while making up facts or generating toxic content. Therefore, GPT-3 was fine-tuned using RLHF to improve its ability to follow user instructions. To develop InstructGPT, the authors of [Ouyang et al. \(2022\)](#) followed a three-stage process presented in Figure 2.3: Supervised fine-tuning (SFT), Reward Modeling (RM), and Reinforcement Learning (RL).

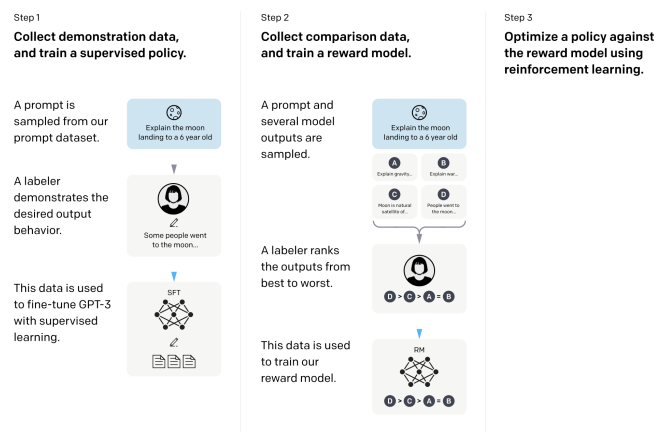


Figure 2.3: InstructGPT 3-step training diagram, from [Ouyang et al. \(2022\)](#)

Firstly, human labelers developed a dataset where, for a given input prompt, the labelers would write a preferred output. These inputs ranged from summarization, question-answer, and others

input types. GPT-3 was fine-tuned on this dataset. Secondly, labelers were given a set of model outputs for a given prompt input, and their goal now was to create a dataset where those outputs were sorted by the the labelers preference of the most appropriate response to the input prompt. With such dataset, GPT-3 was fine-tuned with a regression head that returned a scalar for a combination of model outputs for the same input. The last stage consisted of fine-tuning the SFT model using PPO. For each episode, the policy model (SFT model) would generate an output for a random input prompt, which then would be given to a value function, the RM model, to compute the estimated reward and loss. PPO would then optimize the SFT model policy weights in order to improve further output generation based on the user instructions described in the input prompt given. Equation 2.27 shows the loss function for the RM:

$$Loss_{RM} : \quad loss(\theta) = -\frac{1}{\binom{K}{2}} E_{(x, y_w, y_l) \sim D} [\log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l)))] \quad (2.27)$$

where  $r_\theta(x, y)$  is the scalar output of RM given a prompt  $x$  and a corresponding completion  $y$  with parameters  $\theta$ ,  $y_w$  is the preferred completion out of  $y_w$  and  $y_l$ , and  $D$  is the comparison dataset.

As for GPT-4, reported in OpenAI et al. (2023), it consists of a 100 trillion parameter GPT model trained on publicly available data (such as online resources) and licensed data. GPT-4 followed the same pre-training goal as previous GPT versions and was fine-tuned with RLHF just like InstructGPT.

## 2.2.2 Prompt Engineering

In the context of generative language models, a prompt is the input text that is given to such models, and prompt engineering consists in modifying its structure and content Chen et al. (2023). As research shows, such engineering choices can impose substantial changes to the output given by models, such as in Lu et al. (2022) and Webson and Pavlick (2022).

The systematic design and optimization of prompts to guide LLMs to generate the desired outputs, a more formal definition for prompt engineering, has become a research area with methodologies and established best practices Chen et al. (2023). This section will explain such best practices and methodologies, from the basics to more advanced ones.

### 2.2.2.1 Basic practices

Providing clear and precise instructions and using role-prompting techniques can prompt the generative model to replicate a specific persona with unique qualities. According to the prompt engineering definition Chen et al. (2023), triple quotes are handy when the prompt comprises multiple components or when we are quoting something.

**One-Shot and Few-Shot prompting** There are some essential best practices to remember when using the generative process, such as reusing the exact prompt multiple times for sampling. Additionally, One-Shot (1S) or Few-Shot (FS) Prompting [Chen et al. \(2023\)](#) [Brown et al. \(2020\)](#) is another helpful technique. With 1S prompting, the user gives as input one demonstration/example to the LM, including a question and answer. Then, give the LM the real question, similar to the one from the demonstration, followed by an answer trigger for the LM to fill in the answer. 1S is a specific case of FS where the number of demonstrations (K) is 1. For example, a 2S prompt could be as follows:

**Solve the following arithmetic questions:**

**Q:** The odd numbers in this group add up to an even number: 4, 8, 9, 15, 12, 2, 1.

**A:** The answer is False.

**Q:** The odd numbers in this group add up to an even number: 17, 10, 19, 4, 8, 12, 24.

**A:** The answer is True.

**Q:** The odd numbers in this group add up to an even number: 17, 9, 10, 12, 13, 4, 2.

**A:**

**Temperature and Decoding Sampling** The last basic good practice that [Chen et al. \(2023\)](#) mentions is to control the LM's temperature and decoding sampling. Although this is not a technique to adjust the prompt, it was included in this section to simplify the structure of this section.

The temperature of a language model plays a crucial role in determining the level of creativity or randomness in its output [Chen et al. \(2023\)](#). A lower temperature value often results in more predictable samples, while a higher temperature value leads to more creative and diverse samples [Isaranontakul and Kreesuradej \(2023\)](#). The temperature hyper-parameter is typically applied in the softmax layer, which is responsible for scaling the unnormalized probabilities of the logits using exponential functions [Frei and Kramer \(2023\)](#). By increasing the temperature, the logits approach zero, resulting in higher softmax scores for less likely tokens and lower scores for more likely tokens. This way, the model is more creative and unpredictable. The impact of temperature on the model's output was observed as early as 1985 [Ackley et al. \(1985\)](#), where a sigmoid function was used to illustrate its effect at temperature values of 0.25, 1, and 4. For instance, a temperature value of 0.25 led to significantly higher derivatives near  $x = 0$ , compared to no temperature effect ( $T = 1$ ) and higher temperature values ( $T = 4$ ).

As for decoding sampling, [Chen et al. \(2023\)](#) used GPT-4 [OpenAI et al. \(2023\)](#), so the sampling method for decoding output sequences is the Nucleus Sampling or Top-p Sampling [Holtzman et al. \(2020\)](#). However, other decoding techniques exist, such as Beam Search [Freitag and Al-Onaizan \(2017\)](#) and Top-K Sampling [Fan et al. \(2018\)](#).

As per a research paper on language models [Freitag and Al-Onaizan \(2017\)](#), the Beam Search algorithm is an effective approach that aims to maximize the probability of the output sequence generated by the model. The algorithm involves selecting the top-n symbols with the highest probability at each decoding step for all open prefixes and then narrowing down the number of

sequences to  $N$  (beam) by selecting the top- $N$  sequences with the highest cumulative probability. This process creates  $N$  new prefixes for the next decoding step. Finally, the sequence with the highest probability is chosen from the final  $n$  sequences.

Based on a study by [Fan et al. \(2018\)](#), top- $k$  sampling involves selecting the top- $k$  symbols with the highest scores from the language model for a given prefix. These symbols are then used to sample subsequent symbols from the probability distribution. Similarly, as explained by [Holtzman et al. \(2020\)](#), nucleus sampling involves sampling symbols with a cumulative distribution mass equal to or greater than a threshold parameter  $p$ . In both methods, the probability distribution of previously sampled symbols is used for subsequent sampling.

In comparison to beam search, top- $k$  sampling has been shown to generate higher quality outputs and avoid repetitive phrases that can occur with beam search due to overfitting on the training data, as noted in [Fan et al. \(2018\)](#) and [Holtzman et al. \(2020\)](#).

Nucleus sampling outperforms top- $k$  sampling due to its ability to dynamically adapt to the probability distribution of each symbol at every step. There are various scenarios where the probability distribution of the next token is either too flat or too peaked, with a highly concentrated probability mass in its head over a large number of tokens. A small  $k$  with top- $k$  sampling may generate overly generic outputs for a flat distribution. In contrast, a large  $k$  may lead to more noise by sampling inappropriate tokens for a peaked distribution. Nucleus sampling overcomes this issue by using a threshold  $p$  that considers the fluctuation in the number of candidates based on the distribution of the next token [Holtzman et al. \(2020\)](#).

### 2.2.2.2 Advanced Prompting Methodologies

In [Chen et al. \(2023\)](#) it is also mentioned other prompting techniques besides the ones mentioned above, which they called Advanced Prompting Techniques. The ones explained in this section are Chain-of-Thought, Self-consistency, Generated Knowledge Prompting, and Retrieval Augmentation.

**Chain-of-Thought (CoT)** Chain-of-Thought (CoT) was first introduced by [Wei et al. \(2022\)](#). It combines few-shot prompting with improved LLM performance for tasks requiring arithmetic, common sense, and symbolic reasoning. This novel approach creates a prompt with a triplet structure consisting of input, CoT, and desired output. In contrast, standard few-shot prompting only has a binomial structure with input and output, which is shown in Figure 2.4. According to the paper, using standard few-shot prompting for reasoning or arithmetic tasks leads to poor results.

Humans follow a systematic approach to solving arithmetic or reasoning problems. The CoT approach aims to enable LLMs to do the same. Rather than simply returning the final answer, LLMs using CoT would also provide a thought process, a chain of thought, leading up to the final answer.

In a later study, Zero-shot CoT (0S-CoT) was introduced by [Kojima et al. \(2023\)](#). Unlike its predecessor, 0S-CoT does not rely on any FS reasoning demonstrations on the prompt. Instead,

it adds the phrase "Let's think step by step" to the end of the prompt to retrieve the reasoning. Although other phrases were tested, this one achieved a higher accuracy in generating correct answers. The proposed 2-stage process of OS-CoT involves the extraction of reasoning from the first prompt, followed by the augmentation of the previous prompt with that reasoning. Only then does the model return the answer, which requires a different answer trigger depending on the type of answer needed, whether numerical or multiple-choice.

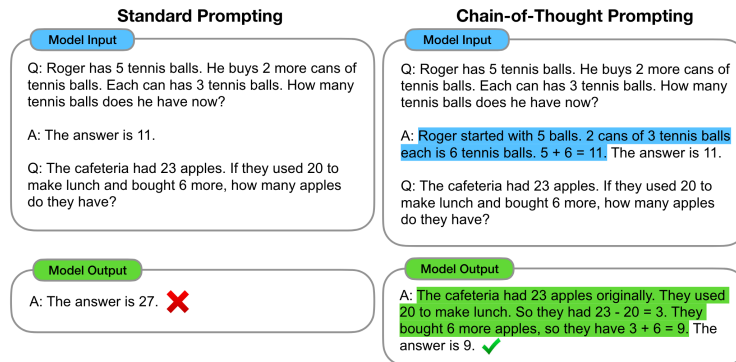


Figure 2.4: Chain-of-Thought summary, from Wei et al. (2022)

**Self-consistency** Self-consistency, as proposed by Wang et al. (2023), posits that when multiple methods produce the same result, the most frequently occurring answer is likely the correct one. Leveraging this concept, we can employ an unsupervised approach to FS-CoT by generating various outputs with their corresponding reasoning and answers and then aggregating the reasoning paths based on their answer. Ultimately, the final answer is chosen based on consistency, including factors such as the number of votes or reasoning paths pointing towards it. Contrary to CoT, which uses a greedy decoding setting to sample the answer from an LM, self-consistency instead generates several answers and samples from those with the most consistent answers. A diagram of such a prompting technique is depicted in Figure 2.5.

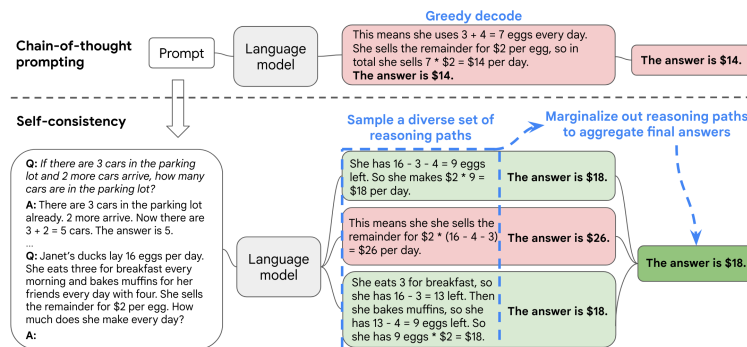


Figure 2.5: Self-Consistency relative to FS-CoT prompting, from Wang et al. (2023)

**Generated Knowledge Prompting (GKP)** GKP, a prompting strategy developed by Liu et al. (2022), enhances the ability of inference models to perform various NLP tasks by providing them with prior knowledge for a given question. This knowledge aids the inference model in answering the question accurately. The generation of knowledge in GKP involves a few-shot approach.

The strategy consists of two primary steps: knowledge generation and its use to augment the original question before prompting it to an inference model. Unlike traditional methods, GKP does not require an external knowledge base to provide prior knowledge. Instead, a generative model produces knowledge based on the question in a few-shot prompting strategy. A schematic of GKP is presented in Figure 2.6.

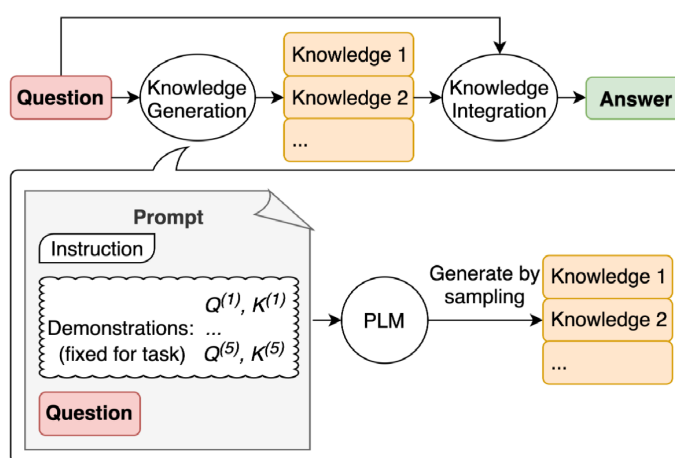


Figure 2.6: Generated Knowledge Prompting summary, from Liu et al. (2022)

The generative model is trained using human-written examples in a few-shot manner. Once trained, a question from the dataset is provided to the model, which returns a sequence corresponding to the knowledge. A sampling process is carried out, retrieving 20 knowledge statements for each question. In the second step of GKP, the questions from the dataset are augmented with the corresponding knowledge, and these questions are then presented to an inference model. For each question, 20 predictions are made, out of which the one with the highest score is selected, and the corresponding knowledge is chosen as the selected knowledge.

Researchers Wu et al. (2023) have explored the use of Few-shot CoT in Knowledge-Augmented Deep Learning (KADL), which integrates external knowledge into deep learning models. Interestingly, their approach utilizes pre-trained Language Models (PLMs) as a knowledge base, allowing for knowledge retrieval without needing an external KB. The authors suggest using CoT prompting to leverage the PLMs potential for knowledge retrieval, making their approach distinct from typical KADL pipelines.

**Retrieval-Augmented Generation (RAG)** In their publication, Chen et al. (2023) discusses a novel prompting technique called Retrieval Augmentation, which addresses one of the challenges

prompt engineering aims to solve: hallucinations. In the context of Natural Language Generation (NLG), hallucinations refer to nonsensical or contextually inaccurate content generated by the LM [Ji et al. \(2023\)](#), a problem that arises when the LM lacks sufficient training data or fails to identify specific patterns while generating output sequences [Chen et al. \(2023\)](#).

One effective way to overcome this issue is to provide the LM with external knowledge. Although there are existing studies that enhance LM performance through fine-tuning and modifying its algorithms and architecture, the work developed in [Lewis et al. \(2020\)](#) tackles exactly that. This work consisted in augmenting a user's prompt or query,  $x$  by retrieving the most relevant chunks of text or documents,  $z$ , which were externally given, to improve the accuracy of the generated token sequence,  $y$ .

Regardless of what token-generation model is used, the innovation of this work comes before that, with the development of a Retriever which, together with the generative model, is optimized during training so that the selection of documents is optimized for a given user's query. There are two main RAG models, however, Equation 2.28 is more generalized as it selects the best documents for each token to be generated.

$$RAG_{Token-Model} : \quad p_{RAG-Token}(y|x) \approx \prod_{i=1}^N \sum_{z \in \text{top-}k(p(\cdot|x))} p_{\eta}(z|x) p_{\theta}(y_i|x, z_i, y_{1:i-1}) \quad (2.28)$$

The two main components of this equation are the retriever,  $p_{\eta}(z|x)$ , and the generative component,  $p_{\theta}(y_i|x, z_i, y_{1:i-1})$ . Regarding the Retriever, its architecture consists of a bi-encoder that computes a latent variable space comprised of dense contextual embeddings for both the user's query,  $q(x)$ , and for the documents,  $d(z)$ . To compute such latent space of contextual embeddings, BERT is used. More specifically, the bi-encoder architecture is shown in Equations 2.29 and 2.30 for the document encoder and query encoder, respectively.

$$\mathbf{d}(z) = \text{BERT}_d(z) \quad (2.29)$$

$$\mathbf{q}(x) = \text{BERT}_q(x) \quad (2.30)$$

Details on this transformer-based encoder model can be seen in Section 2.3.3.2. The dot product between the bi-encoder outputs is computed, measuring the similarity between the document embeddings and the user's prompt embeddings. Such dot products are then scaled using the exponential function, thus giving more emphasis to higher similarities. The formula for such computations can be seen in Equation 2.31.

$$p_{\eta}(z|x) \propto \exp\left(\mathbf{d}(z)^{\top} \mathbf{q}(x)\right) \quad (2.31)$$

Such original RAG models were trained for specific tasks, such as Question-Answering (QA) with specific external files.

### 2.2.3 Privacy Disclosure and Realism

This section will consider, as seen in previous sections, the state-of-the-art review made for the healthcare domain, developed in [Murtaza et al. \(2023\)](#). However, as seen before, the developments made for such a domain, which is the same as this work, are quite few.

This background section of SDG starts with Section 2.2.3.1, where the concept of privacy disclosure is explored, as well as metrics that can be used in an NLP application. The second part, which explores the concept of realism of synthetic samples, is detailed in Section 2.2.3.2. Such a concept can be divided into resemblance and utility. The first measures how similar real and synthetic samples are from a statistical point of view, for instance. The second explores several training and testing schemes to measure how beneficial synthetic samples are from a model performance perspective.

#### 2.2.3.1 Privacy disclosure

As per the findings of [Murtaza et al. \(2023\)](#), early studies had a simplistic view that no privacy disclosures occurred with SD. However, it has been proven that this perspective is too naive. In particular, it has been observed that partially synthetic datasets, which do not modify all feature values from the actual data, carry a higher risk of privacy disclosures.

Regarding privacy disclosures, [Murtaza et al. \(2023\)](#) explains that there are two types of potential breaches: membership disclosure and attribute disclosure. Presence Attacks, or membership disclosure, refer to the attacker's ability to determine an individual's identity with a certain level of confidence from the SD. One can perform such attacks to assess membership disclosure and identify synthetic data points that match real ones. The greater the number of true positives, the higher the presence/membership disclosure risk. Attribute disclosure, conversely, involves inferring actual feature values by examining a subset of the SD. This happens when generative models are over-fitted, and the feature's value interval is very narrow around the actual values of the features.

Concerning metrics, [Murtaza et al. \(2023\)](#) categorizes them into integrated privacy preservation metrics and post hoc privacy preservation metrics. Integrated metrics are incorporated into electronic health records (EHRs) synthesis, ensuring that privacy preservation is considered. There are two types of integrated metrics: wrapper methods and embedded methods. Wrapper methods run parallel to the synthesis process, producing transformations like noise injection or gradient clipping. Embedded metrics are embedded into the generative models' loss function. Post-hoc metrics, on the other hand, are used after the synthesis process and include various nearest-neighbor and distance metrics (e.g., Hellinger, Euclidean, and Hamming). Additionally, synthetic data outliers can lead to privacy breaches, so an outlier similarity metric measures how similar synthetic outliers are to their real counterparts.

As shown in works such as Guan et al. (2021) and Ive et al. (2020), privacy disclosure for a text data granularity can be done using a post-hoc setting with metrics such as BLEU, ROUGE, TER, and Perplexity. Regarding the work developed during this dissertation, the main focus will be on explaining BLEU. Metrics such as ROUGE and Perplexity will also be explained. Regarding the ROUGE metrics, only ROUGE-N and ROUGE-L will be detailed considering the relevancy to this work, especially for Chapter 3, the Literature Review.

**BLEU** Introduced in Papineni et al. (2002), the BLEU score was initially developed for MT to more easily assess candidate translations relative to reference translations. Such candidates and references can be either singular sentences or documents with several sentences. Human evaluations of MT weigh several factors, such as adequacy, fidelity, and fluency. BLEU, firstly, uses n-grams up to N to compute a modified precision metric. Precision is computed by dividing the number of n-grams from the candidate translation that appears in the reference by the total number of n-grams in the candidate translation.

The problem with this precision is that it can give an evaluation of 1 to candidates that repeat the same n-grams present in the reference several times. To handle this problem, the number of n-grams used to compute the precision is clipped by the total number of times that n-gram appears in the reference. This is the modified n-gram precision. The equation of the modified n-gram precision for possibly several candidates and references can be seen in Equation 2.32.

$$p_n = \frac{\sum_{C \in \{Candidates\}} \sum_{n-gram \in C} Count_{clip}(n-gram)}{\sum_{C' \in \{Candidates\}} \sum_{n-gram' \in C'} Count(n-gram')} \quad (2.32)$$

Adequacy is measured using modified unigram precision, while fluency is measured by modified n-gram precision for N greater than 1. Because n-grams with lower  $n$  have a modified n-gram precision higher relative to higher values of  $n$ , the authors proposed using the average logarithm of the modified n-gram precision with uniform weights,  $w_n = 1/N$ , equivalent to the geometric mean.

Consequently, modified n-gram precision penalizes candidate translation lengths greater than reference translations. However, such a metric is not robust to candidates shorter than the references, especially candidates very short. Therefore, a brevity penalty (BP) is used, where for a given candidate and a set of references, the candidate will be penalized if its length is not the exact same as any of the reference's length. Equations 2.33 and 2.33 represent the BP and the BLEU score calculation, respectively:

$$Penalty_{Brevity} : \quad BP = \begin{cases} 1, & \text{if } c > r, \\ e^{(1-r/c)}, & \text{if } c \leq r. \end{cases} \quad (2.33)$$

$$BLEU_{score} : \quad BLEU = BP \cdot \exp \left( \sum_{n=1}^N w_n \log p_n \right) \quad (2.34)$$

**ROUGE** Introduced in [Lin \(2004\)](#), the Recall-Oriented Understudy for Gisting Evaluation (ROUGE) consists of several metrics developed to evaluate summaries by comparing a computer-generated summary to ideal ones developed by humans. Those measures are ROUGE-N, ROUGE-L, ROUGE-W, and ROUGE-S.

ROUGE-N is an n-gram recall metric between the candidate summary and the reference ones. Comparing ROUGE-N to BLEU, while BLEU measures the amount of matching n-grams relative to the total n-grams in the candidate, ROUGE-N does that but relative to the n-grams in the reference documents. When several references are considered, the pair candidate-reference with the highest ROUGE-N score will be considered the final score between the candidate and all the references. This is applicable for all ROUGE metrics, not only ROUGE-N. Equation 2.35 shows how to compute ROUGE-N for considering a single reference document, while Equation 2.36 shows how the final ROUGE-N score is computed considering each ROUGE-N score computed considering a given candidate,  $s$ , and a specific reference document,  $r_i$ , from a set of references  $R$ .

$$ROUGE - N = \frac{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count_{match}(gram_n)}{\sum_{S \in \{ReferenceSummaries\}} \sum_{gram_n \in S} Count(gram_n)} \quad (2.35)$$

$$ROUGE - N_{multi} = \arg \max_i ROUGE - N(r_i, s) \quad (2.36)$$

ROUGE-L, instead of using n-grams, uses the longest ordered subsequence of unigrams to measure matches or overlaps between candidates and references. Such sequences are known as Longest Common Subsequences (LCSs). ROUGE-L is an F-measure of LCS, considering both precision and recall. Such F-measure expression can be seen in Equation 2.41. The precision and recall metrics considering the LCSs at the sentence level can be seen in Equations 2.37 and 2.38, respectively, where  $n$  is the length of the documents  $Y$  and  $m$  the length of the documents  $X$ .

$$R_{lcs \text{ sentence-level}} : \quad R_{lcs} = \frac{LCS(X, Y)}{m} \quad (2.37)$$

$$P_{lcs \text{ sentence-level}} : \quad P_{lcs} = \frac{LCS(X, Y)}{n} \quad (2.38)$$

When measuring ROUGE-L at the summary level or document level, the LCS of the entire document consists of the union of LCS between a reference sentence and a candidate summary for all reference sentences. Such adaptations of Equations 2.37 and 2.38 from sentence-level to document-level can be seen in Equations 2.39 and 2.40 for recall and precision, respectively.

$$R_{lcs} \text{document-level} : R_{lcs} = \frac{\sum_{i=1}^u LCS_{\cup}(r_i, C)}{m} \quad (2.39)$$

$$P_{lcs} \text{document-level} : P_{lcs} = \frac{\sum_{i=1}^u LCS_{\cup}(r_i, C)}{n} \quad (2.40)$$

$$F_{\beta} : F_{\beta} = \frac{(1 + \beta^2) \cdot \text{precision} \cdot \text{recall}}{\beta^2 \cdot \text{precision} + \text{recall}} \quad (2.41)$$

ROUGE-L contains two disadvantages. First, it only considers the main in-sequence order, which can lower the score to shorter sequences. Second, for two candidate sequences with the same LCS length, it doesn't distinguish the candidate with fewer words between ordered words in the LCS. To tackle this second limitation, ROUGE-W, which uses weighted LCSs (WLCSs), remembers the length of consecutive overlaps, where consecutive matches are given higher scores than non-consecutive matches.

### 2.2.3.2 Realism

As noted by Murtaza et al. (2023), realism in electronic health records (EHRs) comprises two key dimensions: resemblance and utility. Resemblance refers to the statistical similarity between real and generated EHRs, while utility pertains to the practical usefulness of synthetic data. A closer examination of realism metrics reveals two categories: quantitative and qualitative.

**Resemblance** When evaluating the quality of a synthetic dataset, we can use qualitative metrics such as expert inspection and visual plausibility. Expert inspection involves having one or more experts review the dataset to assess its plausibility in structure, value ranges, patterns, constraints, and correlations. However, this process can become tedious with large synthetic datasets and is subjective to the expert's knowledge, skill set, and willingness to validate the data.

Quantitative metrics can be divided into univariate and multivariate metrics. Moment-based metrics, such as mean and variance, fall under the univariate category. In contrast, correlations, covariances, principal component analysis (PCA), and other metrics are part of the multivariate category and examine the relationships between features in real and synthetic datasets. On the other hand, visual plausibility involves using data visualizations such as histograms and heatmaps,

as well as condensed projections like PCA or T-SNE to compress high-dimensional data into a lower-dimensional latent space that can be easily visualized in 2 or 3 dimensions.

**Utility** When it comes to measuring the effectiveness of a synthetic dataset, there is no one-size-fits-all solution. The usefulness of the dataset depends on the specific task for which it is being used. However, one commonly used method for measuring utility is the Performance Agreement. More popularly, this involves comparing the results of one ML model instance trained and tested on real data (TRTR) and with a model instance trained on synthetic data and tested on real data (TSTR). Other settings such as TRTS and TSTR have also been deployed. Utility would then be measured, for example, by how close the TRTR and TSTR evaluation results are. Training on real and synthetic data and testing on real data (TRSTR) is also done, where in this case we want to see if an augmented dataset improves performance relative to TRTR [Libbi et al. \(2021\)](#).

Any performance metric can be used depending on the problem being addressed (classification, regression, or unsupervised learning). Another method is Response Agreement, which involves testing two instances of a machine learning model on the same test examples. Augmented Test is yet another approach, where performance metrics are compared between two instances of a model, one trained only on real data and the other trained on both real and synthetic data. If the latter outperforms the former, it suggests that the synthetic data is valuable. More methods are available.

## 2.3 Narrative Extraction

Prior to discussing the narrative extraction pipeline, it is essential to highlight the significance of data acquisition and annotation. While this report will not delve into the former (which is covered in Synthetic Data Generation), the annotation process is paramount. Annotation frameworks must be versatile enough to be applicable across various fields within a language while also accommodating multiple languages. Additionally, the diversity of annotations is crucial [Santana et al. \(2023\)](#).

Moving on to the narrative extraction pipeline, the authors of [Santana et al. \(2023\)](#) summarized a five-step process, including Pre-processing and parsing, Identification and Extraction of Components, Linking Components, Representation, and Evaluation.

### 2.3.1 Pre-processing and Parsing

Lexical and syntactical tasks are involved in preparing and analyzing narratives. Lexical tasks involve pre-processing techniques such as sentence segmentation, tokenization, text cleaning, and normalization (including removing stopwords, stemming, and lemmatization). Meanwhile, syntactical tasks encompass Part-of-Speech (POS) tagging, Parsing (Chunking and Dependency), and Coreference resolution.

To achieve effective results, it is essential to clean and normalize the text through lexical tasks. These tasks play a crucial role in building high-performing models. On the other hand, syntactical

tasks help us understand the meaning of sentences and larger pieces of text by analyzing the grammar class of all words, taking their context into account, and inferring relationships between them.

### 2.3.1.1 Sentence segmentation

As per the findings of [Palmer \(2007\)](#), sentence segmentation involves identifying sentence boundaries determined by the punctuation marks that separate them. This process is also known as sentence boundary detection, sentence boundary disambiguation, or sentence boundary recognition. In [Jurafsky and Martin \(2009\)](#) sentence segmentation is also called sentence tokenization. Ultimately, the aim is to break down text into more digestible chunks and sentences by accurately identifying where one sentence ends and another begins.

While this task may appear simple, it can be quite complex for languages as rich as English. Punctuation marks such as periods, exclamation points, question marks, and ellipses can serve as sentence boundaries or not. Exclamation and interrogation marks are ambiguous, as well as periods, although the latter is less ambiguous than the former punctuation marks [Jurafsky and Martin \(2009\)](#). For example, the period character can either be considered a sentence boundary marker or an abbreviation marker, such in Mr. or Inc. In fact, in this previous example the period is even more ambiguous because it acts as both sentence boundary and abbreviation marker. Also, two sentences separated by a period may be considered a single sentence if the period could be replaced by a comma. The distinction between these situations is crucial for tasks such as POS tagging [Palmer \(2007\)](#). In general, either in rule-based or ML approaches to sentence segmentation, when performing disambiguation the methods first decide if the period is part of a word and then if it is a sentence boundary marker [Jurafsky and Martin \(2009\)](#). To aid in this pre-processing task, either hand-built or machine learned abbreviation dictionaries can be used to disambiguate periods [Kiss and Strunk \(2006\)](#).

**Neural-based Sentence Segmentation** In this regard, sentence segmentation using ML techniques can either be performed as a text classification task ([Wicks and Post \(2021\)](#)) or a sequence tagging task ([Iosifov et al. \(2020\)](#)). As for the sequence tagging approach to sentence segmentation, an encoder (such as BERT) receives the input tokens and returns contextualized embeddings to then be used to predict a binary tag for each token, representing either the last token of a sentence or not [Iosifov et al. \(2020\)](#). In [Wicks and Post \(2021\)](#), a context window is considered to the left and to the right of a period, for example, and the model predicts if such marker is a sentence boundary one or an abbreviation marker, for example [Wicks and Post \(2021\)](#). The architecture's scheme is shown in Figure 2.7:

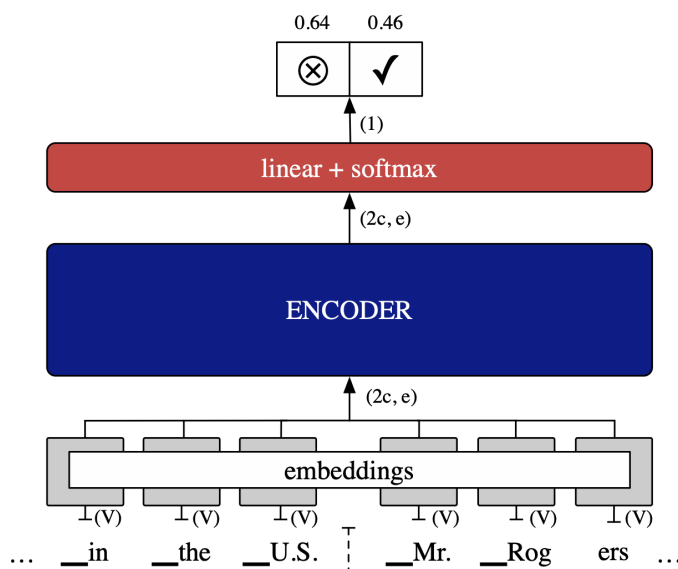


Figure 2.7: A neural-based binary sentence tokenization architecture, from [Wicks and Post \(2021\)](#)

### 2.3.1.2 Text Cleaning

Text cleaning can include removing numbers, punctuation, accents, and common words with little to no use to NLP downstream tasks, commonly known as stop words [Santana et al. \(2023\)](#).

**Python tools** Programming languages such as Python have built-in packages with functions that enable implementing a simple text-cleaning pipeline where tasks such as converting all characters to lowercase or removing all numbers and punctuation. Some tools enable cleaning tasks such as dealing with clitic contractions, meaning expressions such as "I'd like..." or "You're going to love it!" where they should be spelled out as "I would like..." and "You are going to love it!".

**Stopwords** As for stopwords, in [Ladani and Desai \(2020\)](#), such a topic is explored. A stopword is a word with low discriminative power used in languages such as English to connect words with high discriminative power [Ladani and Desai \(2020\)](#). In European Portuguese, examples of stopwords are "a", "de", "e", and "em". Not only that, but stopwords are also words that do not have meaning, do not have predictive capability, and have a high frequency of occurrence in the language [Ladani and Desai \(2020\)](#).

There are two main types of stopword: generic and domain-specific. A generic stopword is a language-specific word used in all areas, including the aforementioned ones. A domain-specific stopword is similar to a generic stopword. However, it is used specifically in a certain area, such as education or sports. There are several benefits in removing stopwords from a corpus. In [Ladani and Desai \(2020\)](#) are mentioned four benefits, those being (1) reducing a corpus by nearly 35-45%, (2) as the corpus decreases, so does the training and testing time of NLP models, (3) while

improving time efficiency it also increases the performance in Information Retrieval (IR) and Text Classification (TC) tasks, and (4) increased data mining applications proficiency and exactness.

Although stopwords removal has these benefits, such a task cannot always be implemented in text-cleaning pipelines. As mentioned above, stopwords removal has the benefit of improving IR and TC tasks. This process can also be implemented for NLP tasks such as auto-tag generation (such as sequence tagging). However, stopwords removal should not be done when the task requires using all words in a language, such as machine translation, language modeling, text summarization, question-answering, and sentiment analysis [Ladani and Desai \(2020\)](#).

As for stopwords identification techniques, there are two types: static and dynamic approaches [Ladani and Desai \(2020\)](#). A static approach to identifying stopwords involves using a fixed list of words relative to a specific language. On the other hand, a dynamic approach to stopwords identification consists of using rules and/or statistics to create a list of stopwords based on a given text corpus. In dynamic approaches, there are rule-based techniques, techniques based on Zipf's Law, and term-based random sampling. Rule-based dynamic techniques involve using stopwords rules and patterns in a given text corpus. As shown in [Piantadosi \(2014\)](#), word frequencies, being one of the most basic properties in a human's communicative system, can be well modeled by a simple power law, Zipf's Law. Zipf's Law applied to word frequencies states that the  $i$ th most frequent word has a frequency  $f(r)$  that scales according to Equation 2.42:

$$\text{ZipfLaw} : \quad f(r) \propto \frac{1}{r^\alpha} \quad (2.42)$$

An equivalent definition of Zipf's law for word frequency was also stated in [Ladani and Desai \(2020\)](#). It was stated that "Given a large sample of words used, the frequency of any word is inversely proportional to its rank in the frequency". This means that the most frequent word in a given corpus has rank 1, for example. There are several techniques to compute such word frequency on a given text corpus, such as Term Frequency (TF), Normalized Term Frequency, Inverse Document Frequency (IDF), and TF-IDF. TF-IDF consists of considering the frequency of a word  $i$  in a document  $j$  (TF) and the frequency of documents with  $i$ ,  $df(i)$ , on the entire set of documents  $N$  (IDF). Equations 2.43, 2.44, and 2.45 show how to compute TF, IDF, and TF-IDF, respectively:

$$\text{TermFrequency} : \quad TF_{i,j} = \frac{n_{i,j}}{\sum_k n_{i,j}} \quad (2.43)$$

$$\text{InverseDocumentFrequency} : \quad IDF_i = \log \left( \frac{N}{df(i)} \right) \quad (2.44)$$

$$\text{TermFrequency} - \text{InverseDocumentFrequency} : \quad TF - IDF_{i,j} = TF_{i,j} \cdot IDF_i \quad (2.45)$$

where  $n_{i,j}$  in the frequency of  $i$  in  $j$ .

**Regular Expressions** One of the highlights regarding text standardization in computer science is regular expressions, commonly shortened to regex [Jurafsky and Martin \(2009\)](#). Regex is a language used to specify string patterns to be searched in a document or a set of documents, a collection. Regex is used in every word processor and text processing tool [Jurafsky and Martin \(2009\)](#). A regex search function will search the entire corpus and return all patterns matching the defined regex. As such, regex can be used to search for certain patterns we don't want in the text, such as URLs, mentions, hashtags, and special characters.

Let's consider we want to remove a European Portuguese stopword such as "de". Using regex language, the following regex can almost perfectly return all "de" patterns in the text:

$$/ [\text{^a-zA-Z}] [\text{dD}] \text{e} [\text{^a-zA-Z}] /$$

In this regex,  $[\text{dD}] \text{e}$  considers patterns with "de" and "De",  $[\text{^a-zA-Z}]$  to the left and right of  $[\text{dD}] \text{e}$  make sure that the patterns "de" and "De" found are not embedded within another word such as "debaixo" or "Depressa". This regex still has a problem, but it works just fine for exposing purposes.

In fact, working with regex usually means several tests until all desired patterns in the text are returned. From this try-and-error process, the goal is to minimize the error rate of the regex. This error rate has two components: precision and recall [Jurafsky and Martin \(2009\)](#). In other words, when working with regex, we want to minimize the number of false positives and false negatives returned.

### 2.3.1.3 Text Tokenization

Text normalization, discussed in Section 2.3.1.4, also includes the task of text tokenization or word tokenization [Jurafsky and Martin \(2009\)](#). However, here word tokenization will be part of a different section. As defined in [Jurafsky and Martin \(2009\)](#), word tokenization consists of segmenting running text into segregated words, such as sentences, paragraphs, and documents. Two categories of word tokenization methods exist: top-down and bottom-up tokenization [Jurafsky and Martin \(2009\)](#). Top-down tokenization rules are applied to implement a tokenization algorithm that aims at a certain standard. Bottom-up methods use character-level statistics to break up words into subwords. In bottom-up approaches, for example, word tokenization does not always return actual words but rather parts of words. Regardless of the segmentation returning words, subwords, or characters, the units given by such algorithms are called tokens.

**Top-down (rule-based) tokenization** When tokenizing running text, several considerations must be made to complete the task successfully. Punctuation is an example, where it could be interesting to consider it a separate token when it is a sentence boundary; however, in other contexts, it is better to consider it part of another token, such as abbreviations like "Ph.D." [Jurafsky and Martin \(2009\)](#). Special characters and numbers for measures or dates might be useful to be kept together instead of being segregated into different tokens. For example, money quantities (\$36.58), dates (18/05/2024), and medical measurements (GFR 9ml/min). In fact, as mentioned in [Jurafsky and](#)

**Martin (2009)**, numbers have an additional complexity when considering different writing patterns of different languages. In English, while commas are used to separate every three digits before the decimal digits and a period is used as the decimal mark, in European languages such as Portuguese, a space separates the digits and the comma as the decimal marker. Tokenizers can also expand clitic contractions, explained in Section 2.3.1.2. For NLP tasks such as named entity recognition, having multiword expressions as a single token may also be performed **Jurafsky and Martin (2009)**, such as "Washington D.C." or "Vila Nova de Gaia".

As mentioned before, top-down tokenization need to follow a certain standard, which then guides rules to tokenize running text. A common tokenization standard is called Penn Treebank tokenization, developed by the Linguistic Data Consortium (LDC) **Jurafsky and Martin (2009)**. Such standard states that clitic contractions must be separated as a different token, hyphenated words must be kept together in the same token, and punctuation markers must be divided as separate tokens. For example, the word "We'd" would be segregated in two tokens, "We" and "'d".

Tokenization needs to be efficient considering that it is required for any NLP task. Therefore, such pre-processing step needs to be efficient, and for that reason, rule-based tokenizers are deterministic methods that use regex **Jurafsky and Martin (2009)**.

For languages where there are no spaces that bound words and where characters-level tokens are not enough, such as Japanese and Thai, word segmentation algorithms are based on Neural Networks **Jurafsky and Martin (2009)**. Chinese, on the other hand, can be used in NLP tasks with character-level tokens **Jurafsky and Martin (2009)**.

**Bottom-up tokenization** While rule-based approaches apply word tokenization and, in Chinese is used character tokenization, bottom-up approaches use the corpus available to determine what will the tokens be **Jurafsky and Martin (2009)**. Using rule-based tokenizers have a disadvantage that is significant in language processing, which is that word tokenization is not able to tokenize words it has not seen in the training corpus **Jurafsky and Martin (2009)**, leading to the usage of a special token, [UNK]. On the other hand, character-level tokenization does not run into this problem; however, in languages such as English or Portuguese, such tokens are meaningless. For example, with word tokenization, if the training corpus has the words "low", "new", "newer" but not "lower", then if in the test corpus "lower" appears, the token used will be [UNK] **Jurafsky and Martin (2009)**. This does not make sense because the morpheme "er" adds to the token "new" the same meaning as to the token "low".

To overcome the unknown word problem, modern tokenizers are capable of generating tokens for words and for parts of words either with or without meaning. Such units, morphemes or not, are called subwords **Jurafsky and Martin (2009)**. Morphemes are detailed in Section 2.3.1.4. Considering the example above, with such a subword tokenizer that is capable of learning words, morphemes, and other arbitrary subwords, when encountering the word "lower,." The tokenizer would be able to return two morphemes instead of [UNK], one of which a word: "low" and "er". In fact, even if the tokenizer did not know of the morpheme "er", still it would not return [UNK], but rather individual characters **Jurafsky and Martin (2009)**.

Generally, bottom-up tokenizers have two components in their scheme: token learner and token segmenter. The token learner will determine a set of tokens, the vocabulary, based on a raw training corpus. The token segmenter will tokenize a raw test sentence based on the token vocabulary learned by the token learner [Jurafsky and Martin \(2009\)](#). There are two algorithms that are widely used: byte-pair encoding ([Sennrich et al. \(2016\)](#)) and unigram language modeling ([Kudo \(2018\)](#)). Unigram language modeling can also be referred to as SentencePiece tokenization [Jurafsky and Martin \(2009\)](#). Because most of the pre-trained LM's use a BPE tokenizer, the unigram language modeling tokenizer will not be detailed.

Regarding **Byte-Pair Encoding (BPE)**, its token learner starts with a vocabulary of all characters where all words are represented as a sequence of characters, plus a special token that represents the end of a word. In [Jurafsky and Martin \(2009\)](#) it is used "\_" while in [Sennrich et al. \(2016\)](#) "." is used as the end-of-word special token. Then, it goes through the training corpus, identifies the two more frequently adjacent characters, and creates a merge. This new merge is added to the vocabulary and replaces adjacent occurrences of such characters with the new token. The algorithm goes through this process until a pre-defined number of merges,  $K$ , is reached. The only hyperparameter of BPE is  $K$ , and the final vocabulary consists of the initial character tokens plus the  $K$  merges learned by the BPE token learner [Sennrich et al. \(2016\)](#). The BPE token learner does not merge cross word boundaries and is not considered to improve the algorithm's efficiency [Sennrich et al. \(2016\)](#), an important characteristic of tokenizers mentioned in [Jurafsky and Martin \(2009\)](#).

Given a small example shown in [Sennrich et al. \(2016\)](#) and [Jurafsky and Martin \(2009\)](#), let's consider a training corpus of 18 words with a total of 5 individual words: low\_ (5 times), lowest\_ (2 times), newer\_ (6 times), wider\_ (3 times), and new\_ (2 times). The vocabulary has a total of 11 characters, as shown in Table 2.1:

| corpus             | vocabulary                      |
|--------------------|---------------------------------|
| 5    l o w _       | _, d, e, i, l, n, o, r, s, t, w |
| 2    l o w e s t _ |                                 |
| 6    n e w e r _   |                                 |
| 3    w i d e r _   |                                 |
| 2    n e w _       |                                 |

Table 2.1: Initial state of the current example of BPE

From here, the BPE learner will check that the most frequent pair of adjacent characters is ("e", "r") with an occurrence frequency 9<sup>1</sup>. So, the pair chosen is merged, creating the new token "er", which increments the BPE vocabulary (sometimes referred to as vocab) and replaces all occurrences of the mentioned pair, as shown in Table 2.2:

<sup>1</sup>Notice that the pair ("r", "\_") is tied with a frequency of 9. When this happens, either one can be chosen.

| corpus |               | vocabulary                          |
|--------|---------------|-------------------------------------|
| 5      | l o w _       | _, d, e, i, l, n, o, r, s, t, w, er |
| 2      | l o w e s t _ |                                     |
| 6      | n e w er _    |                                     |
| 3      | w i d er _    |                                     |
| 2      | n e w _       |                                     |

Table 2.2: Second state of the current example of BPE

In the second iteration, the pair ("er," "\_") is the most frequent. Therefore, the token "er\_" is added to the vocab and replaces all occurrences in the training corpus, as shown in Table 2.3:

| corpus |               | vocabulary                               |
|--------|---------------|------------------------------------------|
| 5      | l o w _       | _, d, e, i, l, n, o, r, s, t, w, er, er_ |
| 2      | l o w e s t _ |                                          |
| 6      | n e w er_     |                                          |
| 3      | w i d er_     |                                          |
| 2      | n e w _       |                                          |

Table 2.3: Third state of the current example of BPE

Then, the pair ("n", "e") is the most frequent. Therefore, the new merger "ne" will be made, therefore appending the new token to the vocab and doing the appropriate replacements in the training corpus, as shown in Table 2.4:

| corpus |               | vocabulary                                   |
|--------|---------------|----------------------------------------------|
| 5      | l o w _       | _, d, e, i, l, n, o, r, s, t, w, er, er_, ne |
| 2      | l o w e s t _ |                                              |
| 6      | ne w er_      |                                              |
| 3      | w i d er_     |                                              |
| 2      | ne w _        |                                              |

Table 2.4: Forth state of the current example of BPE

Considering a  $K = 8$ , five more iterations of the BPE's token learner are required. The process will be rigorously the same as the one showcased in this example. The final iterations of the BPE token learner are summarized in Table 2.5:

| merge      | current vocabulary                                                       |
|------------|--------------------------------------------------------------------------|
| (ne, w)    | _, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new                        |
| (l, o)     | _, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo                    |
| (lo, w)    | _, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low               |
| (new, er_) | _, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_       |
| (low, _)   | _, d, e, i, l, n, o, r, s, t, w, er, er_, ne, new, lo, low, newer_, low_ |

Table 2.5: Last 5 iterations of the current example of BPE token learner, given the hyperparameter  $K = 8$

As for the BPE's token segmenter receives a sentence with words segmented as characters to match the initial vocab of the BPE token learner. This way, the token segmenter will apply the merges learned in the same order they were learned, meaning the frequencies in the test corpus will not be considered, only the one on the training corpus (Sennrich et al. (2016), Jurafsky and Martin (2009)).

#### 2.3.1.4 Text Normalization

Another pre-processing step mentioned in Santana et al. (2023) is text normalization, in which lemmatization and stemming were mentioned. Stemming and Lemmatization are applied to observe discrepancies between either equal or similar words Santana et al. (2023). According to Jurafsky and Martin (2009), text normalization, or word/token normalization, consists of transforming words/tokens to follow a certain standard format.

**Case Folding** Although it was mentioned in Section 2.3.1.2, one of the most basic token normalization procedures is case folding Jurafsky and Martin (2009). Case folding stands for mapping all tokens to lowercase characters, meaning that the tokens "Paulatinamente" and "spaCy" would be mapped to "paulatinamente" and "spacy", respectively. Case folding is beneficial to certain NLP tasks such as the ones related with IR and speech recognition, however for tasks such as sentiment analysis, MT, and information extraction case can be more helpful Jurafsky and Martin (2009).

It is also important to mention that pre-processing pipelines that use bottom-up text tokenization approaches such as BPE may not need further text normalization Jurafsky and Martin (2009).

**Stemming** Stemming involves a computational process that removes affixes from inflected words to obtain the root word. Stemmers are designed to reduce the size of the document index, resulting in improved computational efficiency. This technique also helps reduce the time and space complexity of natural language processing tasks Jabbar et al. (2020).

Morphemes are the most minor units of language that cannot be further divided into meaningful parts. They are combined to form other words with meaning. For instance, in "disagreement," "disagree" is a meaningful morpheme, and "ment" is a meaningless morpheme. However, when combined, they form a meaningful word. The stem is the base morpheme to which other morphemes, such as affixes, are attached. Conversely, the root is similar to the stem but consists of

only two morphological units. In the case of "disagreement," the root is "agree" [Jabbar et al. \(2020\)](#).

There are two main categories of stemming methods: linguistic-based and computational-based. Linguistic-based stemmers extract the stems by following grammatical rules, while computational-based stemmers utilize statistical or non-statistical computations to extract stems [Jabbar et al. \(2020\)](#).

One widely known example of a rule-based stemmer is the Porter Stemmer, developed by Martin Porter in 1951 [Porter \(1980\)](#). On the other hand, ML-based stemmers belong to the category of statistical stemmers. The best-performing stemmer currently available is a linguistic-based technique introduced by [Graën et al. \(2018\)](#), as shown by [Santana et al. \(2023\)](#).

**Lemmatization** As defined by [Bergmanis and Goldwater \(2018\)](#), lemmatization involves identifying an inflected word's base form, or lemma. For instance, "swimming" would be reduced to "swim". This is beneficial in a practical manner, for example, when searching on a search engine for swimming pools, as we expect that both "swim pool" and "swimming pools" queries return exact or very similar results. Such inflected word variations are very common in morphologically complex languages [Jurafsky and Martin \(2009\)](#). For example, in European Portuguese, it is expected that the several verbal forms of the irregular verb "ir", such as "vou", "vão", "foram", "irão", are reduced to their lemma, "ir".

The most sophisticated lemmatizers use morphological parsing [Jurafsky and Martin \(2009\)](#), where morphology is the study of how words are constructed from smaller components, the morphemes. As seen in stemming, words are built using the most meaningful morpheme, the stem, together with the other type of morpheme, the affixes. Therefore, such lemmatization methods receive a word such and parse it considering its morphemes, returning the lemma [Jurafsky and Martin \(2009\)](#). While this may seem straightforward, lemmatizers face two challenges: determining the lemma of unfamiliar or unseen words and disambiguating inflections based on context [Bergmanis and Goldwater \(2018\)](#). As shown in [Bergmanis and Goldwater \(2018\)](#), such problems can be handled with a character-level attentional encoder-decoder recurrent architecture, using as context  $N$  characters to the left and right of the word to be lemmatized. Special tokens are used to define a word boundary, the beginning and ending of the word to be lemmatized.

#### 2.3.1.5 Part-of-speech (POS) tagging

Attributing syntactic tags, such as nouns, verbs, adverbs, and adjectives, to the words in a text is the process of POS tagging [Santana et al. \(2023\)](#). POS tagging (POST), also known as grammatical tagging, consists of automatically assigning part-of-speech tags to words in a given sentence [Chiche and Yitagesu \(2022\)](#). POS tagging is one of the most NLP areas that has been addressed by research, being a crucial basis for other NLP tasks such as machine translation, word sense disambiguation (WSD), and question answering parsing [Chiche and Yitagesu \(2022\)](#).

**Parts of speech** Parts of speech have been defined in the Universal Dependencies tagset [de Marneffe et al. \(2021\)](#) and fall into two main categories shown in Table 2.6 (excluding the last division): closed class and open class [Jurafsky and Martin \(2009\)](#). Closed class parts of speech are those with a fixed membership (relative to open class parts of speech), such as adpositions (prepositions and postpositions). They usually are words called function words, such as "of", "and", and "it", which commonly are short, occur frequently, and have grammatical significance in structuring a sentence [Jurafsky and Martin \(2009\)](#). As for open class parts of speech, where new words in classes such as nouns or verbs keep being created and borrowed [Jurafsky and Martin \(2009\)](#).

| Tag   | Description                                                                                              | Example                        |
|-------|----------------------------------------------------------------------------------------------------------|--------------------------------|
| ADJ   | Adjective: noun modifiers describing properties                                                          | red, young, awesome            |
| ADV   | Adverb: verb modifiers of time, place, manner                                                            | very, slowly, home, yesterday  |
| NOUN  | words for persons, places, things, etc.                                                                  | algorithm, cat, mango, beauty  |
| VERB  | words for actions and processes                                                                          | draw, provide, go              |
| PROPN | Proper noun: name of a person, organization, place, etc.                                                 | Regina, IBM, Colorado          |
| INTJ  | Interjection: exclamation, greeting, yes/no response, etc.                                               | oh, um, yes, hello             |
| ADP   | Adposition: marks a noun's spacial, temporal, or other relation                                          | in, on, by, under              |
| AUX   | Auxiliary: helping verb marking tense, aspect, mood, etc.,                                               | can, may, should, are          |
| CCONJ | Coordinating Conjunction: joins two phrases/clauses                                                      | and, or, but                   |
| DET   | Determiner: marks noun phrase properties                                                                 | a, an, the, this               |
| NUM   | Numeral                                                                                                  | one, two, 2026, 11:00, hundred |
| PART  | Particle: a function word that must be associated with another word                                      | 's, not, (infinitive) to       |
| PRON  | Pronoun: a shorthand for referring to an entity or event                                                 | she, who, I, others            |
| SCONJ | Subordinating Conjunction: joins a main clause with a subordinate clause such as a sentential complement | whether, because               |
| PUNCT | Punctuation                                                                                              | ;,()                           |
| SYM   | Symbols like \$ or emoji                                                                                 | \$, %                          |
| X     | Other                                                                                                    | asdf, qwfg                     |

Table 2.6: The 17 parts of speech in the Universal Dependencies tagset in English ( [de Marneffe et al. \(2021\)](#)), from [Jurafsky and Martin \(2009\)](#)

**Tagging** POS taggers consist in sequence-to-sequence (seq2seq) models where for each input token  $x_i$  in an input sequence  $\mathbf{x}$  a part-of-speech tag  $y_i$  will be assigned, returning an output sequence  $\mathbf{y}$  with the same length as  $\mathbf{x}$  [Jurafsky and Martin \(2009\)](#), as shown in Figure 2.8. POST is a disambiguation task considering that a word can have more than one POS tag [Jurafsky and Martin \(2009\)](#), meaning that the context is very important to infer a word's POS tag. Using an example from [Jurafsky and Martin \(2009\)](#), the word "book" can either be a verb, "book a flight", or a noun, "hand me that book". Such ambiguities is what POS taggers aim to resolve.

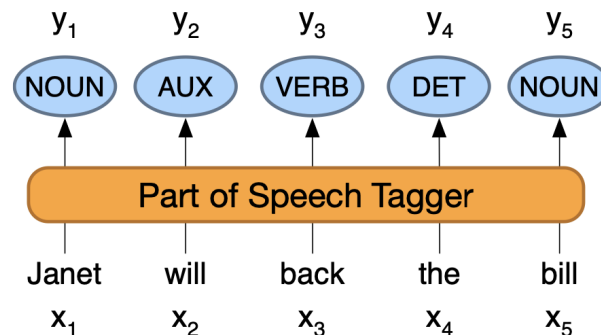


Figure 2.8: A schematic of the POST task where a token sequence is processed by an encoder together with a classification head (POS tagger) that predict a tag for each input token, from [Jurafsky and Martin \(2009\)](#)

Manual POST is very inefficient from several perspectives, such as time and human resources. Three main types of POST settings exist: rule-based, probabilistic-based, and transformational-based [Chiche and Yitagesu \(2022\)](#). Rule-based POS taggers use manually defined linguistic rules to constrain the assignment of POS tags to words. For example, in Portuguese, a word that comes before an adjective can be tagged as a noun, while in English is the opposite. Probabilistic approaches consider several contexts with correctly labeled examples in a text corpus to compute a probability for the POS tag of a certain word in a sentence. Transformational-based approaches combine both linguistic rules and probability computation [Chiche and Yitagesu \(2022\)](#).

To develop efficient POST solutions, researchers have been working with ML and DL techniques [Chiche and Yitagesu \(2022\)](#). According to [Santana et al. \(2023\)](#), the current state-of-the-art (SOTA) POS tagger was developed by [Bohnet et al. \(2018\)](#), which is a DL solution that uses bidirectional recurrency with BiLSTM.

### 2.3.2 Identification and Extraction of Components

While pre-processing and parsing text are crucial steps, the first main task lies in extracting the individual components of a narrative, such as participants, time expressions, events, and spatial components. The second main task, linking components, will not be covered.

#### 2.3.2.1 Participants

In narratives, the individuals or entities participating in the events are called participants. They are named entities and can be detected through Named-Entity Recognition (NER). NER focuses on identifying participants like people, organizations, and places, but it can also recognize other entities such as time, dates, and quantities [Santana et al. \(2023\)](#). NER can be accomplished through four approaches: rule-based, unsupervised learning, supervised learning, and deep learning [Santana et al. \(2023\)](#).

According to [Alshammari and Alanazi \(2021\)](#), NER tags usually follow a certain annotation scheme. These schemes can be the following: IO, IOB, IOE, IOBES, BI, IE, and BIES.

The IO tagging scheme is the most simple one, where the tag "I" is used to identify a participant while the tag "O" refers to normal words that don't represent any participant. The IOB tagging scheme, or BIO, has been adopted by the Conference on Computational Natural Language Learning (CoNLL) [Tjong Kim Sang and Buchholz \(2000\)](#). The "B" tags the initial token of the participants, "I" for the participant tokens after the first one, and "O" for the non-participant tokens in the sequence. The IOE tagging scheme is the same as IOB. However, the beginning tag is replaced by an ending tag for the participant. IOBES combines the same participant tags as IOB and IOE, with a "S" tag for single token participants. The BI tagging scheme is similar to IOB, with the difference that normal tokens tagged with "O" also receive the beginning and inside tags "B-O" and "I-O". As for IE, it also adds the end and inside tags for non-participants to the IOE labels. BIES is very similar to IOBES, however now non-participants also have beginning, inside, and ending tags - "B-O", "I-O", "E-O". For single non-participant tokens, the tag now becomes "S-O".

### 2.3.2.2 Time

Narratives often reference specific times, such as "last year" or "yesterday". Temporal Tagging can be broken down into two tasks: time expression extraction and normalization [Santana et al. \(2023\)](#). Temporal extraction involves identifying temporal expressions (TIMEX) within the text, which can be achieved through NER. However, temporal normalization can be more complex, requiring pinpointing the exact point in time being referenced and, for example, determining the specific day referred to by "yesterday" or the year/week referred to by "this year" or "last week" [Santana et al. \(2023\)](#).

**TIMEX Detection** Temporal expressions are text spans that refer to absolute points in time, relative times, and durations [Jurafsky and Martin \(2009\)](#). Absolute TIMEX can be mapped directly to a date and/or a time of the day. Relative TIMEX maps to specific moments in time like absolute TIMEX but with another time reference. Durations refer to time spans in several possible granularities, such as seconds, minutes, halves, hours, days, centuries, and so on [Jurafsky and Martin \(2009\)](#). Such TIMEX categories are summarized in Table 2.7:

| Absolute                | Relative                 | Durations               |
|-------------------------|--------------------------|-------------------------|
| April 24, 1916          | yesterday                | four hours              |
| The summer of '77       | next semester            | three weeks             |
| 10:15 AM                | two weeks from yesterday | six days                |
| The 3rd quarter of 2006 | last quarter             | the last three quarters |

Table 2.7: Examples of absolute, relative and durational TIMEX, from [Jurafsky and Martin \(2009\)](#)

TIMEX are grammatical constructions with a main text span as the component that enhances the existence of a TIMEX. Such text spans are called temporal lexical triggers, and they are the

heads of the TIMEX. Such temporal triggers can nouns, proper nouns, adjectives and adverbs, as shown in Table 2.8:

| Part of Speech            | Examples                                                                        |
|---------------------------|---------------------------------------------------------------------------------|
| Noun                      | morning, noon, night, winter, dusk, dawn                                        |
| Proper Noun               | January, Monday, Ides, Easter, Rosh Hashana, Ramadan, Tet                       |
| Adjective                 | recent, past, annual, former                                                    |
| Adverb                    | hourly, daily, monthly, yearly                                                  |
| Specialized time patterns | 8:00, 12/2/00, 1994, 1960s                                                      |
| Time noun/adverb          | now, today, yesterday, tomorrow                                                 |
| Number                    | 3 ("He arrived at 3."), fifth ("the fifth of June"), Sixties ("in the Sixties") |

Table 2.8: Examples of temporal lexical triggers, from [Ferro et al. \(2005\)](#)

In fact, the existence of such lexical triggers is what makes a TIMEX markable [Ferro et al. \(2005\)](#). Also, for a text span to be considered the trigger of a TIMEX, its referent must enable mapping such TIMEX either on a timeline or enable a relative time inference (past, present, future) [Ferro et al. \(2005\)](#). There are text expressions that relate semantically to time but are considered non-triggers. For example, "earlier", "next", "later", "beforehand", and "previously" are examples of non-trigger adverbs [Ferro et al. \(2005\)](#). More details on what is considered a lexical trigger can be found in [Ferro et al. \(2005\)](#).

Because words such as "yesterday" are the lexical trigger that identify the TIMEX category, a complete TIMEX consists on the their phrasal projections: noun phrases, adjective phrases, and adverbial phrases [Jurafsky and Martin \(2009\)](#).

So, the task at hand is detecting complete TIMEX considering their markable lexical triggers. Below there is an example of such task using TIMEX3 tags taken from [Jurafsky and Martin \(2009\)](#):

A fare increase initiated <TIMEX3> **last week** </TIMEX3> by UAL Corp's United Airlines was matched by competitors over <TIMEX3> **the weekend** </TIMEX3>, marking the second successful fare increase in <TIMEX3> **two weeks** </TIMEX3>.

As for approaches to detecting TIMEX spans, in [Jurafsky and Martin \(2009\)](#) are mentioned two main ones: rule-based approaches and sequence-labelling approaches.

Works such as in ([Chang and Manning \(2012\)](#), [Strötgen and Gertz \(2013\)](#), [Chambers \(2013\)](#)) used cascades of regex to extract parts of speech, lexical triggers, and classes. SUTime, a regex-based tool for TIMEX detection developed in [Chang and Manning \(2012\)](#) and used in [Chambers \(2013\)](#), uses to detect Durations the following regex, for example:

$$/(\backslash d+)[-\backslash s](\$TEUnits)(s)?([-\backslash s]old)?/$$

Regarding sequence-tagging approaches, these use tagging schemes such as IOB and a classifier is trained with either embeddings (contextual or not) followed by an encoder or classic features extracted from the text, such as POS and lexical triggers [Jurafsky and Martin \(2009\)](#).

**Temporal Normalization** After considering how to detect TIMEX in a corpus, it is crucial to define certain attributes to characterize such temporal expressions. In other words, the TIMEX detected need to be mapped to a certain point in time or duration [Jurafsky and Martin \(2009\)](#). Such difference is shown in Table 2.7. Such normalization follows the ISO 8601 standard for encoding temporal values ( [International Organization for Standards \(ISO\) \(2004\)](#)). In fact, there are two main TIMEX annotation standards that follow ISO 8601, with some additional content: TIDES TIMEX2 and TimeML which uses TIMEX3 to annotate temporal expressions [Strötgen and Gertz \(2013\)](#).

The TIDES annotation standard follows the principle that TIMEX should be tagged "if a human can determine a value for [it]" and if the value of such TIMEX can be derived from the text [Ferro et al. \(2001\)](#). Regarding normalization, the TIMEX2 tags contain several possible attributes to which values can be assigned, shown in Table 2.9:

| Attribute  | Function                                                                | Example                   |
|------------|-------------------------------------------------------------------------|---------------------------|
| VAL        | Contains a normalized form of the date/-time.                           | VAL="1964-10-16"          |
| MOD        | Captures temporal modifiers.                                            | MOD="APPROX"              |
| ANCHOR_VAL | Contains a normalized form of an anchoring date/time.                   | ANCHOR_VAL="1964-10-16"   |
| ANCHOR_DIR | Captures the relative direction/orientation between VAL and ANCHOR_VAL. | ANCHOR_DIR="BEFORE"       |
| SET        | Identifies expressions denoting sets of times.                          | SET="YES"                 |
| COMMENT    | Contains any comments the annotator wants to add.                       | COMMENT="context garbled" |

Table 2.9: TIMEX2 tag attributes, from [Ferro et al. \(2005\)](#)

With this segregation in mind, for a TIMEX such as "early last night", here is an example of what a normalized temporal expression taken from [Ferro et al. \(2005\)](#), following the TIDES TIMEX2 standard looks like:

```
<TIMEX2 VAL="2000-10-31TNI" MOD="START">early last night</TIMEX2>
```

The TimeML standard takes a step further than the TIDES standard by also considering annotations for events, relations between events and TIMEX, and between two events [Pustejovsky et al. \(2003\)](#). Regarding events, these are further detailed in Section 2.3.2.3. More attributes were added to the TIMEX3 annotation scheme relative to TIMEX2 tags. There are four types of TIMEX3 tags: TIME, DATE, DURATION, and SET [Pustejovsky et al. \(2005\)](#). TIME and DATE tags models expressions related with the time of the day and DATE can be thought of as a calendar time. TIME and DATE examples are shown in Table 2.10:

|                | TIME                                     | DATE                           |
|----------------|------------------------------------------|--------------------------------|
| Mr. Teale left | <b>ten minutes to three</b>              | <b>Friday, October 1, 1999</b> |
|                | <b>at five to eight</b>                  | <b>the second of December</b>  |
|                | <b>at twenty after twelve</b>            | <b>yesterday</b>               |
|                | <b>at half past noon</b>                 | <b>in October of 1963</b>      |
|                | <b>at eleven in the morning</b>          | <b>in the summer of 1964</b>   |
|                | <b>at 9 a.m. Friday, October 1, 1999</b> | <b>on Tuesday 18th</b>         |
|                | <b>the morning of January 31</b>         | <b>in November 1943</b>        |
|                | <b>late last night</b>                   | <b>this year's summer</b>      |

Table 2.10: TIMEX3 examples for TIME and DATE tag types, from Pustejovsky et al. (2005)

DURATION explicitly describes a time extent, such as "2 months", "36 hours", or "all night". SET refers to expressions that regularly occur, such as "twice a week" or "once every day" Pustejovsky et al. (2005). One of the TIMEX3 attributes is, therefore, the type of TIMEX. That being said, depending on the tag type the attribute set is different, as we will see further next. Considering the input text used above as an example of the TIMEX detection task, below we can see their respective TIMEX3 annotations:

```
<TIMEX3 id="t2" type="DATE" value="2007W26" anchorTimeID="t1">
last week </TIMEX3>
```

```
<TIMEX3 id=" t3 " type="DURATION" value="P1WE" anchorTimeID="t1">
the weekend </TIMEX3>
```

```
<TIMEX3 id="t4" type="DURATION" value="P2W" anchorTimeID="t1">
two weeks </TIMEX3>
```

The first TIMEX3 annotation refers to a particular week of a certain week. According to the ISO standard, weeks are numbered from 01 to 53, where week 01 is considered the one with the first Thursday of the year Jurafsky and Martin (2009). These weeks are represented with the format YYYY-Wnn International Organization for Standards (ISO) (2004). Notice the attribute *type* corresponds to the TIMEX3 tag type. The attribute *id* is simply the unique identifier of the TIMEX3 annotation so that it can be mentioned for further analysis Pustejovsky et al. (2005), and the attribute *value* is the ISO format applied to the specific week, in this case the 26th week of 2007.

As for the second TIMEX3 annotation, because weekends are treated as DURATIONS the attribute *value* must be a length. The ISO encoding for durations is Pnx, where n is the length and x is a unit. In this case, "the weekend" is considered as one weekend, hence the usage of the normalized duration P1WE. In the third annotation becomes clear why the ISO encoding for the temporal expression "two weeks" is P2W.

That being said, there is a catch. One of the attributes was not mentioned yet on purpose, the

*anchorTimeID*. If we look at the original text with the TIMEX detection, we can see that there is no way of inferring that the temporal expression "last week" refers to the 26th week of 2007. So, how does the TIMEX3 annotation has such information? In Pustejovsky et al. (2005), the authors explain that when a TIMEX3 is underspecified, these are anchored to a fully-specified TIMEX3 annotation that is appended to the ones of a given document. This usually gives the creation date of the document. Therefore, a fourth TIMEX3 annotation is considered, in this example the one below, where its *id* is the same as the one that appears on the other three TIMEX3 in the attribute *anchorTimeID*:

```
<TIMEX3 id="t1" type="DATE" value="20070702"
functionInDocument="CREATION TIME"> July 2, 2007 </TIMEX3>
```

However, when considering a SET tag type, the attributes are different, as shown below:

```
<TIMEX3 tid="t1" type="SET" value="P1W" quant="EACH" freq="3D">
3 days each week </TIMEX3>
```

In a SET annotation, besides the attributes above, we also have the *quant* and the *freq* attributes. In this case, the *value* attribute must collaborate with at least of those two new attributes. *quant* represents the quantifier used, while *freq* represents the frequency Pustejovsky et al. (2005).

Most TIMEX normalizers are rule-based (Chang and Manning (2012), Strötgen and Gertz (2013)), however that does not mean that such task is easy. In fact, it is very difficult because real texts with fully-qualified TIMEX are very rare Jurafsky and Martin (2009). In news articles, for example, most of the temporal expressions are incomplete and are only anchored with the dateline of an article. To such information, we call temporal anchor Jurafsky and Martin (2009). Not only that, but simple expressions such as "the weekend" add some complexity. In the running example, "the weekend" refers to the preceding weekend of the document creation date, however, in other examples, it might be the weekend that proceeds it Jurafsky and Martin (2009).

### 2.3.2.3 Events

Events can be defined as occurrences that happen at a specific time and place and have consequences Santana et al. (2023). Therefore, the event extraction task consists in detecting all event occurrences in pre-processed texts Jurafsky and Martin (2009). The following text was extracted from Jurafsky and Martin (2009), where in a given text all events are highlighted:

Citing high fuel prices, United Airlines said Friday it has increased fares by \$6 per round trip on flights to some cities also served by lower-cost carriers. American Airlines, a unit of AMR Corp., immediately matched the move, spokesman Tim Wagner said. United, a unit of UAL Corp., said the increase took effect Thursday and applies to most routes

where it **competes** against discount carriers, such as Chicago to Dallas and Denver to San Francisco.

Journalists often use a set of questions called 5W1H to define events [Santana et al. \(2023\)](#). The history of event detection dates back to 1998 with the development of the Topic Detection and Tracking (TDT) project, which specifically focused on detecting and tracking news stories based on events. This part of the project became known as the First Story Detection (FSD) [Allan et al. \(1998\)](#).

In the English language, most of the events are verbs, as well as most events occur after a verb [Jurafsky and Martin \(2009\)](#). In the example above, however, there are some exceptions. The events "the move" and "the increase" are introduced by a noun phrase (NP). Some verbs often fail to introduce events, such as phrasal and light verbs [Jurafsky and Martin \(2009\)](#). An example of a phrasal verb occurs in the example above in "took effect". A light verb is a verb that has little to almost no meaning, such as make, take, and have. Such verbs are associated with their direct object noun, which ends up being the event [Jurafsky and Martin \(2009\)](#). An example of this is in the expression "have a meeting", where "have" is a light verb and "meeting" is the event.

**Event Extraction** There are several versions of event extraction, depending on the objective in mind. For example, the objective is to extract events and aspectual and temporal characteristics in the **TempEval shared tasks** ([Verhagen et al. \(2009\)](#)). Events can be categorized, for example, as actions, states, reporting events (say, report, tell), and perception events [Jurafsky and Martin \(2009\)](#). Considering the example above, the three occurrences of the event "said" would be classified as (class=REPORTING, tense=PAST, aspect=PERFECTIVE).

Another standard for event extraction was developed by the **Automatic Content Extraction (ACE)** where a terminology and hierarchy for events was defined, categorizing events into eight types and 33 sub-types [Linguistic Data Consortium \(LDC\) \(2005\)](#). In the first stage of identifying events, there are two spans of text relevant to consider: the event extent and the event trigger. The event extent, or scope, is the full sentence in which there is a taggable event characterized, while the event trigger is the word that identifies the existence of an event in a given scope [Linguistic Data Consortium \(LDC\) \(2005\)](#). These event trigger are what determine what is the event type and sub-type in the event scope. Such event types and sub-types are shown in Table 2.11:

| Event Type  | Event Sub-type                                                                                                                       |
|-------------|--------------------------------------------------------------------------------------------------------------------------------------|
| LIFE        | BE-BORN, MARRY, DIVORCE, INJURE, DIE                                                                                                 |
| MOVEMENT    | TRANSPORT                                                                                                                            |
| CONTACT     | MEET, PHONE-WRITE                                                                                                                    |
| CONFLICT    | ATTACK, DEMONSTRATE                                                                                                                  |
| BUSINESS    | START-ORG, MERGE-ORG, DECLARE-BANKRUPTCY, END-ORG                                                                                    |
| TRANSACTION | TRANSFER-OWNERSHIP, TRANSFER-MONEY                                                                                                   |
| PERSONELL   | START-POSITION, END-POSITION, NOMINATE, ELECT                                                                                        |
| JUSTICE     | ARREST-JAIL, RELEASE-PAROLE, TRIAL-HEARING, CHARGE-INDICT, SUE, CONVICT, SENTENCE, FINE, EXECUTE, EXTRA-DITE, ACQUIT, APPEAL, PARDON |

Table 2.11: Event types and sub-types, from [Linguistic Data Consortium \(LDC\) \(2005\)](#)

The events not only are classified at the type and sub-type levels, but they also have some characteristics as in the TempEval shared tasks standard. Using ACE-like annotations, the events also have arguments. Event arguments can either be participants or attributes [Linguistic Data Consortium \(LDC\) \(2005\)](#). These participants are entities that can be tagged as being involved in the event. For each event type and sub-type there are specific participants that can be involved. For example, in a MARRY event sub-type the participants are the people who are getting married, while in the event DIE the participants can only be the entity that killed and the one getting killed. Regarding the attributes, these can either be specific to the event or general attributes. Specific attributes are arguments specific to a certain event, while general attributes are arguments that appear on all events. For example, for the event type JUSTICE there is a specific attribute for the crime committed (CRIME-ARG), or the event JUSTICE.SENTENCE with an attribute related with the sentence that was levelled against a DEFENDENT-ARG after conviction (SENTENCE-ARG). General attributes are the place where the event occurred (PLACE-ARG) and when it occurred (TIME-ARG). TIME-ARG are formatted to TIMEX2 annotations. There are some rules for such arguments such as the shared argument rule, where it is stated that if two events share the same text expression as an argument, then such argument will be part of both event annotations.

Another event annotation scheme, a standard used in biomedical NLP, is called **GENIA**. The GENIA event annotation relies on two ontologies: event ontology and term ontology [Kim et al. \(2008\)](#). Roughly speaking, the event ontology can be seen as the vocabulary of the events, while the term ontology gives the arguments that are used to describe events [Kim et al. \(2008\)](#).

A similar structure for the term ontology hierarchy appear in [Kim et al. \(2008\)](#). This event ontology is based on the Gene Ontology (GO) event ontology [Ashburner et al. \(2000\)](#), with some additions such as the event type `Artificial_Process`.

As for modeling considerations, in [Jurafsky and Martin \(2009\)](#), it is stated that supervised machine learning, more specifically sequence tagging with IOB tagging scheme (explain in Section 2.3.2.1), is performed where event classes and characteristics are assigned to their respective multi-class classifiers. The models can either follow classical methodologies with feature-based models or more advanced and modern approaches with neural-based encoders such as BERT (ex-

plained in Section 2.3.3.2) [Jurafsky and Martin \(2009\)](#). A set of several features commonly used in feature-based approaches to event extraction are shown in Table 2.12:

| Feature                    | Explanation                                                           |
|----------------------------|-----------------------------------------------------------------------|
| Character affixes          | Character-level prefixes and suffixes of target word                  |
| Nominalization suffix      | Character-level suffixes for nominalizations (e.g., -tion)            |
| Part of speech             | Part of speech of the target word                                     |
| Light verb                 | Binary feature indicating that the target is governed by a light verb |
| Subject syntactic category | Syntactic category of the subject of the sentence                     |
| Morphological stem         | Stemmed version of the target word                                    |
| Verb root                  | Root form of the verb basis for a nominalization                      |
| WordNet hypernyms          | Hypernym set for the target                                           |

Table 2.12: Features commonly used in classic feature-based approaches to event extraction, from [Jurafsky and Martin \(2009\)](#)

#### 2.3.2.4 Space

The space aspect of written stories challenges identifying geographical details from the text, closely related to Geographical Information Retrieval (GIR). Even after detecting spatial references, linking them to actual locations remains a problem to be solved. An excellent example provided by the authors of [Santana et al. \(2023\)](#), the sentence "John, the magician, pulled a rabbit out of a hat at the show in Edinburgh this year," highlights this issue. It is still being determined which show is being referred to and where it took place, given that there may be several shows in Edinburgh. Additionally, "in Edinburgh" can indicate a specific neighborhood or the broader region of Edinburgh, Scotland.

When exploring the use of space in narratives, two significant challenges emerge. Firstly, spatial context inevitably exists in any narrative, even if not explicitly mentioned as a narrative component. This can lead to difficulty identifying locations when space is merely a narrative setting rather than a central feature [Santana et al. \(2023\)](#). For instance, a text may mention "the Capitol" and immediately bring to mind Capitol Hill in Washington, D.C. However, when considering the entire narrative, that location could be incorrect, and the actual location could be the capital city of Panem in "The Hunger Games" film and book series. Secondly, resolving coreferences of common references in lengthy texts can pose another problem [Santana et al. \(2023\)](#). Unlike coreference resolution of spacial components such as locations in long documents, the ability to coreference common references such as house and room can be challenging.

### 2.3.3 Modeling Techniques

Throughout the literature, several modeling techniques have been used to accomplish all the narrative extraction tasks mentioned so far, such as models to create lemmatizers, participant identification (or NER), temporal reasoning, relation extraction, and so on. Some of the techniques and architectures described in Section 3.1 were also used or inspired to develop such models.

Therefore, here will be added other techniques that weren't described yet, such as the BERT architecture [Devlin et al. \(2019\)](#).

### 2.3.3.1 Conditional Random Fields (CRF)

In [Lafferty et al. \(2001\)](#), the authors take advantage of Maximum entropy Markov models (MEMMs), which consists of a generative model that computes a distribution probability of tokens using conditional probabilities, to work on a disadvantage it has, which the authors called the Label Bias Problem.

Such a problem consists of transitions leaving a given state compete only against each other at that time step, rather than against all other transitions before in the autoregressive procedure. These transition scores, or conditional probabilities, are normalized at each time step given that the cumulative probability over the current state's distribution is equal to one. The authors call this the "conservation of score mass". Although MEMMs consider the observation sequence, the sequence already generated, to compute conditional probabilities, according to [Lafferty et al. \(2001\)](#), such observations can only affect which states get the mass, but not how much mass to pass on to each state. This causes a problem for transitions less common in the corpus.

Conditional Random Fields, CRF, in the context of sequence labeling applications such as NER, is a model that tackles the label bias problem while keeping the advantages of the MEMM, which were already mentioned. To do so, CRF computes the dependencies between labels in a sequence and incorporates features from both input and neighboring labels. A visual comparison between MEMMs and CRF is shown in Figure 2.9. Considering, for example, a participant extraction task with a IOB tagging scheme, explained in Section 2.3.2.1, a CRF can be trained to understand that the first token of a participant has a B- prefix and the next tokens of such participant can only have a I- prefix.

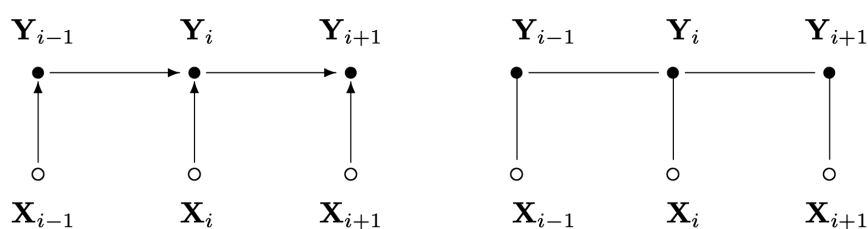


Figure 2.9: MEMMs (left) vs CRF (right), from [Lafferty et al. \(2001\)](#)

### 2.3.3.2 Bidirectional Encoder Representations from Transformers (BERT)

Google researchers developed BERT in [Devlin et al. \(2019\)](#). The idea behind BERT is to have a fixed transformer-based pre-trained LM versatile enough for several NLP downstream tasks, such as classification, NER, and question-answering (QA). GPT [Radford et al. \(2018\)](#) is a model that is able to do just that, however the GPT pre-training procedure included doing autoregressive token

prediction based on the tokens already generated, by the decoder, to the left. BERT, on the other hand, was designed to predict tokens based not only on the tokens to the left but also to the right of the token to be generated [Devlin et al. \(2019\)](#).

BERT's architecture is composed of the Transformer's encoder from [Vaswani et al. \(2017\)](#). Two BERT instances were pre-trained, BERT base and BERT large. The first has 12 encoder layers with 12 attention heads each and a model dimension size of 768 hidden states, while the latter has 24 encoder layers with 16 attention heads and a model dimension size of 1024 hidden states ([Devlin et al. \(2019\)](#)).

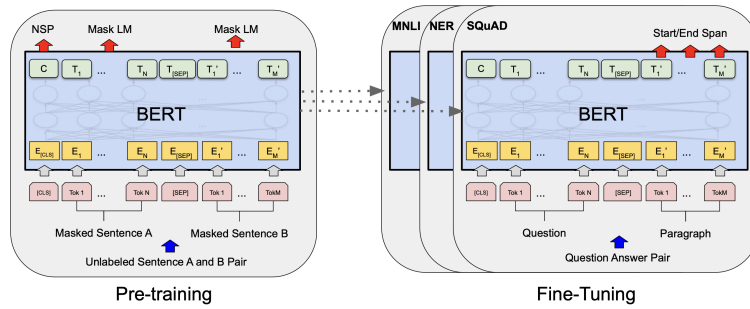


Figure 2.10: BERT pre-training and fine-tuning procedures, from [Devlin et al. \(2019\)](#)

The input sequences are numerical tokens from which token, segment, and positional embeddings are retrieved and fed to the encoder. The input sequence starts with a classification special token, [CLS], then by two sentences, each with a different segment embeddings, separated by a [SEP] special token [Devlin et al. \(2019\)](#).

BERT was pre-trained using a two-step pre-training process involving two unsupervised tasks, simultaneously: Masked Language Modeling (MLM), referred in the literature as a cloze task [Taylor \(1953\)](#) and Next Sentence Prediction (NSP) [Devlin et al. \(2019\)](#).

In order to pre-train BERT in this bidirectional representation fashion to model the token's probability distribution conditionally without the model indirectly having access to the token to be predicted, 15% of the tokens in the input sequence were randomly selected to be predicted and masked with the [MASK] special token. This, however, creates a mismatch between pre-training and fine-tuning because no downstream NLP task has the [MASK] token. So, after selecting 15% of the tokens for prediction, 80% of the time they would be replaced with the mask token. The remaining 20% was equally divided between replacing with a random token or keeping the original token. After that, the final hidden states of the selected tokens given by the encoder would go through a softmax layer for prediction and the cross entropy loss would be back-propagated [Devlin et al. \(2019\)](#). Equation 2.46 reflects the MLM objective, where  $X$  is the input sequence and  $\tilde{X}$  is the same sequence but changed with the [MASK] tokens and  $\mathcal{C}$  is the set of masked tokens in  $X$ .

$$MLM_{objective} : \max_{\theta} \log p_{\theta}(X|\tilde{X}) = \max_{\theta} \sum_{i \in \mathcal{C}} \log p_{\theta}(\tilde{x}_i = x_i|\tilde{X}) \quad (2.46)$$

The second pre-training task employed was Next Sentence Prediction (NSP). Token-level language modeling is not able to capture the relationship between pairs of sentences. NSP is, in fact, a simple binary classification problem where 50% of the times the next sentence in the sequence is, in fact, the actual next sentence, while in the other half of the times it was not. The final hidden states, given by the encoder, of the classification token, [CLS], were used to compute the probability of the sentence after the [SEP] special token being the next sentence of the prior sentence in the input sequence [Devlin et al. \(2019\)](#).

Despite being fine-tuned with pairs of sentences, BERT can be fine-tuned for downstream NLP tasks that require only one sentence, such as tagging. Also, when fine-tuning at the token level, the final token representations given by the hidden states returned by the encoder are given to an output layer. For classification at the sequence level the final representation of the [CLS] special token, the vector  $C$ , is fed to an output layer, such as for sentiment analysis [Devlin et al. \(2019\)](#).

A schematic of both pre-training and fine-tuning approaches to BERT are shown in [Figure 2.10](#).

### 2.3.3.3 Decoding-enhanced BERT with Disentangled Attention (DeBERTa)

In [He et al. \(2021\)](#), a Microsoft research team developed a transformer-based encoder architecture similar to the one developed for BERT, in [Devlin et al. \(2019\)](#). However, there are two main differences, which concern relative and absolute positions of tokens in a certain input sequence. These two differences go by the names of disentangled attention and enhanced mask decoder. It should also be noticed that DeBERTa does not have a NSP objective, just MLM.

Regarding disentangled attention, while in BERT and the original Transformer architecture the positional encodings were combined together with the token embeddings, or the content, in DeBERTa that data is fed and used by the self-attention mechanism separately. According to [He et al. \(2021\)](#), considering token  $i$ , its token embeddings (content) and relative position with token  $j$  can be represented as two vectors,  $H_i$  and  $P_{i|j}$ , respectively. Therefore, the attention score for  $i$  relative to  $j$  can be represented as shown in [Equation 2.47](#).

$$Attention_{score} : A_{i,j} = H_i H_j^\top + H_i P_{j|i}^\top + P_{i|j} H_j^\top + P_{i|j} P_{j|i}^\top \quad (2.47)$$

This representation enhances the fact that the final attention weight can be decomposed into four attention scores, where  $H_i H_j^\top$  represents the content-to-content disentangled matrix,  $H_i P_{j|i}^\top$  the content-to-position,  $P_{i|j} H_j^\top$  the position-to-content disentangled matrix, and  $P_{i|j} P_{j|i}^\top$  the position-to-position. The authors argue that this last disentangled matrix is not too relevant as it only considers the relative positions, thus removed. So, considering  $Q_c$ ,  $K_c$ , and  $V_c$  the Query, Key and Value token embedding vectors, and  $Q_r$  and  $K_r$  the Query and Key relative position vectors, these are calculated with their respective weight matrices as shown in [Equation 2.48](#).

$$\begin{aligned}
Q_c &= HW_{q,c}, & K_c &= HW_{k,c}, & V_c &= HW_{v,c}, \\
Q_r &= PW_{q,r}, & K_r &= PW_{k,r}
\end{aligned} \tag{2.48}$$

Therefore, the final attention score expression is represented as shown in Equation 2.49, where the position-to-position attention score is not considered. The values  $\delta(i, j)$  and  $\delta(j, i)$  represent the relative position bias from token  $i$  to  $j$  and vice-versa. Details on its expression, as well as on the scaling term used to normalize the attention scores, can be seen in He et al. (2021).

$$\text{Attention}_{score} : \tilde{A}_{i,j} = \underbrace{Q_i^c K_j^{c\top}}_{(a) \text{ content-to-content}} + \underbrace{Q_i^c K_{\delta(i,j)}^{r\top}}_{(b) \text{ content-to-position}} + \underbrace{K_j^c Q_{\delta(j,i)}^{r\top}}_{(c) \text{ position-to-content}} \tag{2.49}$$

As for enhanced mask decoder, this concept comprises of the idea that there are certain situations where the relative positions are not enough to decode the masked tokens. While disentangled attention deals with relative positions, it is required to also consider absolute positions. BERT adds absolute positional encodings in the input with the position embeddings. However, because absolute positions might not always be necessary, they are given as additional information after all the transformer encoder block computations and before computing the token's distributions in the softmax layer.

Thus, with the usage of disentangled attention and enhanced mask decoder, DeBERTa is able to calculate attention scores for both token embeddings and relative position encodings separately inside the transformer encoder block, while on the outside, before the softmax computations, it gives absolute position encodings in the form of additional information considering scenarios where relative position encodings are not enough.

### 2.3.3.4 Gradient descent optimization algorithms

To make sure the model's trainable parameters are adjusted to the specific task, either that being a downstream NLP task such as Participant Extraction, Relation Extraction, or other tasks such as MLM or NSP, it is necessary to use algorithms to tune them. For this section, the basic gradient descent variants and the Adam optimizer will be explained based on the overview work developed in Ruder (2016).

Overall, gradient descent algorithms consist of a systematic procedure to minimize an objective function,  $J(\theta)$ , usually called loss function in to context of ML. To achieve that, the model's parameters,  $\theta$ , where  $\theta \in R^d$ , are parameterized by updating them in the opposite direction of the gradient of the loss function for the current state of parameters,  $\nabla_{\theta} J(\theta)$ .

Another important component, and one of the most important hyperparameters of a neural network, is the learning rate,  $\eta$ . The  $\eta$  value adjusts  $\nabla_{\theta} J(\theta)$ , thus controlling how drastic or minimal we want the parameters to be adjusted. For example, when a model is still far from a

local optimum of  $J(\theta)$  it is common sense to have a relatively high  $\eta$ . However, once the model's performance starts reaching a local minimum of  $J(\theta)$ , it is beneficial for  $\eta$  to have relatively lower values as to converge better instead of always staying around it, or even diverge.

Despite the vast amount of gradient descent algorithms, which are especially utilised to optimize neural networks, they can all be used according to one of three variants. Those variants are Batch Gradient Descent, Stochastic Gradient Descent (SDG), and Mini-batch Gradient Descent. The first consists in calculating  $\nabla_{\theta}J(\theta)$  for the entire training dataset. SDG consists in calculating  $\nabla_{\theta}J(\theta)$  for one data sample or instance at a time, for example a row in case of tabular data or a sequence in case of NLP or time-series forecasting. The most common is mini-batch gradient descent, where the gradient is computed for a mini-batch of data instances from the training dataset. Usually, the mini-batch sizes tend to be exponents of base 2, such as 16, 32, or 64. Equations 2.50, 2.51, and 2.52 showcase the three gradient descents.

$$\text{GradientDescent}_{Batch} : \quad \theta = \theta - \eta \cdot \nabla_{\theta}J(\theta) \quad (2.50)$$

$$\text{GradientDescent}_{Stochastic} : \quad \theta = \theta - \eta \cdot \nabla_{\theta}J(\theta; x^{(i)}; y^{(i)}) \quad (2.51)$$

$$\text{GradientDescent}_{Mini-batch} : \quad \theta = \theta - \eta \cdot \nabla_{\theta}J(\theta; x^{(i:i+n)}; y^{(i:i+n)}) \quad (2.52)$$

While Equation 2.50 guarantees a local minimum, Equation 2.51 does not guarantee it, however it can find better local minima but compromises convergence. Mini-batch, on the other hand, takes the benefits of both variants. Also, by considering a mini-batch of data instances, it can also take advantage of the current DL frameworks and GPUs which are capable of handling matrix multiplications. Despite usually being called SDG, this algorithm gives the flexibility of changing the mini-batch size. By using powers of 2, excluding  $2^0$ , despite being commonly called SDG, in practice the variant is mini-batch gradient descent.

As for Adam, or Adaptive Moment Estimation, it uses exponential smoothing to better direct the optimization process. Exponential smoothing is used to calculate the estimates for the mean and uncentered variance of the gradients.

$$\text{Gradient}_t : \quad g^{(t)} = \nabla J(\theta^{(t-1)}) \quad (2.53)$$

So, considering the gradient for a certain optimization step,  $g^{(t)}$ , the first moment and second moment can be computed considering previous gradients in an exponentially decaying faction.

Equation 2.53 emphasizes the gradient expression at time step  $t$ . Equations 2.54 describe how the first and second momentum are calculated, respectively.

$$\begin{aligned} m_t &= \beta_1 m_{t-1} + (1 - \beta_1) g_t \\ v_t &= \beta_2 v_{t-1} + (1 - \beta_2) g_t^2 \end{aligned} \quad (2.54)$$

Equations 2.55 consist of an adjustment made to  $m_t$  and  $v_t$  considering they have bias for values near zero considering they are initialized at zero.

$$\begin{aligned} \hat{m}_t &= \frac{m_t}{1 - \beta_1^t} \\ \hat{v}_t &= \frac{v_t}{1 - \beta_2^t} \end{aligned} \quad (2.55)$$

One of the disadvantages of vanilla gradient descent is that it does not adapt the learning rate for each trainable parameter. With  $m_t$  and  $v_t$ , adaptive learning rates are achieved, following the parameter update expression in Equation 2.56.  $\varepsilon$  is a term used so that divisions by zero do not occur. By default,  $\varepsilon$  is set to  $10^{-8}$ , while  $\beta_1 = 0.9$  and  $\beta_2 = 0.999$ .

$$\theta_{t+1} = \theta_t - \frac{\eta}{\sqrt{\hat{v}_t} + \varepsilon} \hat{m}_t \quad (2.56)$$

### 2.3.4 Evaluation

As pointed out in Santana et al. (2023), as more research has been done and knowledge has been increasing, formalization was needed. This happened not only for narrative extraction components such as identification and extraction of components, as seen in Section 2.3.2, but also for evaluation.

The same authors also mention in their survey how there are two main ways in which narratives can be measured. The first is its evaluation being made considering in-domain criteria and using people as evaluators. The other way is doing that evaluation using metrics such as the F-score. This metric can be seen in Section 2.2.3, more specifically in Equation 2.41.

That being said, it can be also said that narratives can be evaluated in other ways too. A narrative extraction pipeline can be either evaluated as a whole or based on its individual NLP tasks associated (Santana et al. (2023)).

Evaluation procedures made at the narrative level take advantage of the cloze task, defined in Taylor (1953). Such a task was also employed when pre-training BERT, as mentioned in Section 2.3.3.2. Such a task consists of assessing how well a system is proficient in a certain language by having it predict a missing word given a context window of words from which the system can extract information. In this case of narrative-level evaluation, works such as in Chambers and

Jurafsky (2008) and Mostafazadeh et al. (2016) employed such a cloze task. While in Chambers and Jurafsky (2008) a sequence of events (detailed in Section 2.3.2.3) was given and a system had to predict the missing event and dependencies, in Mostafazadeh et al. (2016) the cloze task was employed at the sentence level and not event level. More precisely, in Mostafazadeh et al. (2016) a set of four sentences was given and a model or system had to predict which of two outcomes was the correct one.

As can be seen from the several NLP tasks that exist in terms of extracting components (see Section 2.3.2), each task needs to be evaluated differently. For some tasks the general formula for the F-score may be the standard, but for other tasks that might not be the case. Considering this dissertation's focus is on NER, the evaluation at the task level will be made only for the extraction of participants.

As mentioned before, there are formalizations made to assess several NLP downstream tasks, being NER one of them. Those can be expressed as shared tasks, for instance, as mentioned in Santana et al. (2023). Three of those examples are the CoNLL-2002 and CoNLL-2003 shared tasks, as well as the SemEval 2013 Task 9.1, which were developed in Tjong Kim Sang (2002), Tjong Kim Sang and De Meulder (2003), and Segura-Bedmar et al. (2013), respectively.

All those tasks use as evaluation metric the F-score with  $\beta = 1$ , also called the F-1 score. The difference is in how precision and recall are calculated, and more particularly, what is considered a False Positive (FP) and a False Negative (FN). The reason for this is that the goal of a NER evaluation script is that it can showcase its performance at the entity level. However, because the tagging schemes, such as IOB, are applied at the token level, the outputs are at the token level, not entity level.

Therefore, in Tjong Kim Sang (2002) and Tjong Kim Sang and De Meulder (2003) a FP and FN are calculated considering the concept of a correct prediction, which in this case is not a label of a token correctly predicted, but rather a whole entity matching the ground truth, both in terms of participant type and its span or boundaries. This means it can be a sequence of non-O consecutive tokens that cover the entire span of the participant.

In Segura-Bedmar et al. (2013), however, the concept of what a correct prediction is at the entity level is further developed. The exact match detailed for the CoNLL-2002 and CoNLL-2003 tasks for the SemEval 2013 Task 9.1 is called strict evaluation. However, considering that such a criterion could be considered very rigorous, such task decided to create three more ways of measuring what a correct prediction is. Such ways, or scores, are called exact boundary matching, partial boundary matching, and type matching. These scores, as well as strict evaluation, consider two criteria: boundary and type matching.

Hence, while strict evaluation considers exact boundary and type matching relative to the ground truth, exact and partial boundary matching disregards type matching. As for type matching, the type between the prediction and the ground truth needs to be the same, so there just needs to exist some span overlap between them for a prediction to be considered correct. Such evaluation script, developed in Segura-Bedmar et al. (2013), was further transposed into a Python package,

called *nervaluate*<sup>2</sup>.

Although these scores are quite straightforward, some minor details can make the performance using similar scores differ significantly. For example, let's consider a model which its only objective is to predict a single participant, for example, drugs, using an IOB tagging scheme. Notice that, in such a scenario, strict evaluation and exact boundary matching give the same model performance. If the correct span is "ibuprofen" and the predicted span is "of ibuprofen", the score which will give a lower performance will be partial boundary matching, relative to type matching. This happens because while the former score considers the prediction partially correct, the latter considers it a correct prediction. From a calculation perspective, a partial match is only considered half correct. Precision and Recall formulas using the partial boundary matching score are shown in Equations 2.57 and 2.58, respectively.

$$Precision_{partial} : P = \frac{Correct + 0.5 \times PartiallyCorrect}{TotalPredicted} \quad (2.57)$$

$$Recall_{partial} : P = \frac{Correct + 0.5 \times PartiallyCorrect}{TotalLabeled} \quad (2.58)$$

## 2.4 Summary

This chapter aimed at exploring and explaining the concepts required to understand this dissertation, from more important concepts if SDG such as the GPT models until ChatGPT, Few-Shot Learning (FSL) and RAG, resemblance and utility, to the concepts for narrative extraction such as the distinction between participants, events, TIMEX, encoder architectures such as BERT, and evaluation.

---

<sup>2</sup><https://pypi.org/project/nervaluate>

## Chapter 3

# Literature Review

This section, as mentioned in Section 1.4, has the objective of exploring the most recent and relevant literature available considering the two main components of this dissertation, as mentioned in Chapter 1. The background layed down in Chapter 2 is important to understand such developments in the literature.

With the vast developments the AI scene has been seeing, specially regarding NLP and specifically for participant extraction, the literature found already considers that prior background knowledge, which is quite extensive as seen in Chapter 2, a given for the readers. Hence the importance of such background. In the case of SDG, this is a field that recently has received attention by the research community, as will become evident in this chapter, however very few works actually summarize this fast-growing field of study. Such work was made in the background, thus being important to understand it for this component of the dissertation too.

### 3.1 Outline

This chapter will give emphasis to the literature review made in order to guide the several aspects and decisions required to be made during the practical component of this dissertation. As with the background's structure, in Section 2, this section will also be divided into the main two components of this work, SDG and Narrative Extraction. Literature review for such components is presented in Sections 3.2 and 3.3, respectively.

In Section 3.2, the several papers selected were reviewed considering the model architectures used in the GMs deployed, as well as the usage of prompt engineering techniques to generate SD, finishing with a literature review on privacy disclosure and realism on such works. These reviews can be found in Sections 3.2.1, 3.2.2, and 3.2.3, respectively.

As for the second component, Narrative Extraction, it was mainly divided as shown in Santana et al. (2023). First, the pre-processing and parsing procedures employed used in the papers selected were reviews, as well as the extraction of participants, and evaluation topics of the Narrative

Extraction pipeline. The visualization is a component to which not much resources have been allocated. These topics can be found in Sections 3.3.1, 3.3.2, and 3.3.3, respectively.

## 3.2 Synthetic Data Generation

This section tackles the literature review made for the first main component of this dissertation, Synthetic Data Generation (SDG). Although a search for works in this regard using Portuguese was made, no relevant works were found. Several factors were considered in the search for such works, mainly the search for the most recent papers related to the healthcare sector, and then in a more general domain when few works were found. The lack of literature on clinical or biomedical texts was particularly felt in the most recent development in the NLP field, the usage of LLMs.

This section considers a considerable sample of works. Such papers are repeatedly described in each subsection of Section 3.2. This section focuses on one topic at a time while discussing all the works considered for this literature review. As for the SDG component of the literature review, the paper selection criteria included the SJR (Scimago Journal Ranking) score (when available), the number of citations, and the number of influential citations. Consensus is a great online tool that shows these metrics and uses AI to aid in finding the most relevant works with a higher reputation. Some of these papers, especially those regarding LLM applications, haven't been published, thus only available in an open-source archive, specifically arXiv<sup>1</sup>. Besides that, all of them are published in Q1 or Q2 journals or conferences. We consider LLM applications to be works that use more powerful and well-known decoder-transformer-based models, such as GPT-3.5, GPT-4o, and LLaMa.

Considering the shift found in the literature from 2023 onwards considering the popularity of LLMs, each section in the literature review of SDG has a special subsection reserved for LLM applications, with works developed in 2023 and 2024. This literature review section starts with Section 3.2.1 describing the generative model architectures deployed to generate the data samples. Section 3.2.2 details the several prompting techniques that are schemes given to certain model architectures to generate the new data samples. Section 3.2.3 deals with how privacy concerns are applied in the literature, as well as how the realism, which contains both resemblance and utility, of such synthetic data samples is measured

### 3.2.1 Generative Models

Text-based EHR generative models still need to be explored compared to other data granularities like snapshots or longitudinal data Murtaza et al. (2023). Approaches, such as the original Transformer architecture Ive et al. (2020) and an Encoder-Decoder architecture using LSTM recurrent networks Lee (2018), have shown promise.

---

<sup>1</sup><https://arxiv.org>

Compared to such works that use ML techniques to generate new EHRs, works like [Begoli et al. \(2018\)](#) employed efforts to improve SDG by deploying simple, non-probabilistic, and language modeling approaches. [Begoli et al. \(2018\)](#) leverages literature, KBs, and data analysis to augment a SOAP-like template. This was done using a tool called Synthea [Walonoski et al. \(2017\)](#).

Besides [Lee \(2018\)](#), in [Libbi et al. \(2021\)](#) was implemented an LSTM-based architecture, leading to observed improvements in a downstream NER task. [Libbi et al. \(2021\)](#) also utilized GPT-2, which was also used by [Lu et al. \(2021\)](#). Despite [Libbi et al. \(2021\)](#) doing fine-tuning to GPT-2, the authors of [Lu et al. \(2021\)](#) instead took advantage of KL Divergence ([Kullback and Leibler \(1951\)](#)) and real clinical notes availability to fine-tune a student model on synthetic data generated by a teacher model. KL divergence was crucial to minimize the label probability distribution entropy of the student model relative to the teacher model.

While [Lee \(2018\)](#) and [Libbi et al. \(2021\)](#) used GPT-2, in a recent study, [Peng et al. \(2023\)](#) developed an GM using a GPT-3 architecture and trained on approximately 290M clinical notes to generate synthetic clinical notes using nucleus sampling. The model was named GatorTronGPT.

GAN-based architectures have also been used for generating EHRs, even outside of the text-level granularity, such as for time series [Li et al. \(2023\)](#) [Yoon et al. \(2023\)](#). As for healthcare text generation, [Guan et al. \(2021\)](#) developed a CGAN architecture, which the authors called mtGAN, where the REINFORCE algorithm is used given the token's discrete type. This happens because the discriminator is not able to back-propagate the gradients to the generator considering its discrete token outputs generated, according to [Li et al. \(2021\)](#). Not only that, but REINFORCE also enabled the discriminator to assess non-fully generated sequences by using Monte-Carlo search to generate K finished sequences and back-propagate the results.

In [Li et al. \(2021\)](#), however, all of these different modeling architectures were used. In total, four different GMs were trained on discharge summaries from the 2010 i2b2/VA NLP challenge ([Uzuner et al. \(2011\)](#)), those models being GPT-2, SegGAN, CharRNN, and CTRL. Just like in [Guan et al. \(2021\)](#), SegGAN deals with the non-continuous token nature. CTRL ([Keskar et al. \(2019\)](#)) is a conditional transformer model that uses control codes to adapt the outputs generated, thus having more control over the texts generated.

**LLM Applications** Large Language Models (LLMs), more recently, have also been used in generative applications towards augmenting datasets. Most of the research found focuses on the usage of ChatGPT and the LLMs it comprises, which can be found in [Tang et al. \(2023\)](#), [Ubani et al. \(2023\)](#), [Møller et al. \(2024\)](#), [He et al. \(2024\)](#), [Gilardi et al. \(2023\)](#), and [Dai et al. \(2023\)](#). Some papers didn't explicitly mention which model was used, just mention ChatGPT. In [Møller et al. \(2024\)](#), [He et al. \(2024\)](#), and [Gilardi et al. \(2023\)](#), however, the models used were GPT-4 and GPT-3.5 for the last to papers, respectively. An LLM which is not part of the GPT-family of models used in ChatGPT was used in [Møller et al. \(2024\)](#), which is Llama-2.

Literature on the usage of ChatGPT to generate clinical or biomedical reports to augment training corpus to fine-tune downstream NLP task models is still very scarce. This conclusion was made considering that only one paper was found, specifically the work made in [Tang et al. \(2023\)](#).

An interesting trend was found in the usage of such LLMs. Although in [Tang et al. \(2023\)](#), [Ubani et al. \(2023\)](#), [Møller et al. \(2024\)](#), and [Dai et al. \(2023\)](#) use ChatGPT with either GPT-3.5 or GPT-4 for SDG, in [He et al. \(2024\)](#) and [Gilardi et al. \(2023\)](#) ChatGPT was used as an annotator for data samples. The goal of such LLM applications is not to augment a training dataset with synthetic samples, but rather to annotate existing real samples using their generative capabilities.

As for the papers where such LLMs are used for data generation, such trend was also found where the LLMs are used directly in downstream NLP tasks. The difference, however, is that while in [He et al. \(2024\)](#) and in [Gilardi et al. \(2023\)](#) focused on comparing the annotation capabilities of LLMs against human crowdsourced annotators, in the remaining works, LLMs performing a downstream NLP task were considered baselines for fine-tuned models.

This was made in a 0-shot scheme in [Tang et al. \(2023\)](#), and [Møller et al. \(2024\)](#), while in [Dai et al. \(2023\)](#) ChatGPT was used as a classifier in a 2-shot scheme. Overall, models fine-tuned in an augmented training dataset performed better than the LLMs which generated the SD. In [Møller et al. \(2024\)](#), however, where GPT-4 and Llama-2 were used, on certain text classification tasks the 0-shot performance of such LLMs was superior to that of the fine-tuned model. From all papers which have LLM applications, not only was [Tang et al. \(2023\)](#) the only one which was applied in the clinical or biomedical domains, but also the only one that was applied to NER (and RE). Results showed a much higher performance of fine-tuned models with a full synthetic training corpus relative to the 0-shot performance of ChatGPT.

### 3.2.2 Prompt Engineering

While in [Schick and Schütze \(2021b\)](#) a methodology is developed that combines ICL with fine-tuning (FT), [Frei and Kramer \(2023\)](#) focused solely on ICL to generate synthetic data for NER. They used a FS prompt methodology using GPT-NeoX [Black et al. \(2022\)](#). Special markers are added to the prompt's demonstrations to define the start and ending of a demonstration, as well as markers to delimit narrative components. Each new synthetic data point, in this case, a sentence, is generated using 12S. These authors also applied ICL-only generated data to a downstream NLP task, although prompts were relatively simple with small demonstrations [Schick and Schütze \(2021a\)](#).

While [Lee \(2018\)](#) tested both greedy sampling and beam search as decoding techniques, the authors of [Peng et al. \(2023\)](#) used nucleus sampling instead. Other works don't explicitly mention it. As for temperature regularization, such a technique was employed by [Frei and Kramer \(2023\)](#) to regulate the model's replication of training data.

**LLM Applications** As far as prompt engineering goes, several schemes and complexity levels were used. In these LLM applications, besides the prompt given to the model, ChatGPT enables instructions be given to the LLM on how to behave as, which can also be used to receive external knowledge such as concept explanations. These are called system instructions. The actual prompt is called user instruction.

First the LLM applications focused on data generations will be explored, and then the two papers which used LLMs as annotation generators, such as the works developed in [He et al. \(2024\)](#) and [Gilardi et al. \(2023\)](#). While the purposes for using ChatGPT as a classifier were different between the aforementioned papers and the works developed in [Tang et al. \(2023\)](#), [Møller et al. \(2024\)](#), and [Dai et al. \(2023\)](#), the basis to the LLM applicationn was the same. In [Ubani et al. \(2023\)](#) ChatGPT was not used as baseline. In both scenarios, the LLMs were used for inference without any fine-tuning, but rather ICL.

Considering the LLM applications for data generations, and starting with the most simple prompts, in [Tang et al. \(2023\)](#) and [Ubani et al. \(2023\)](#) a prompt scheme was created with a simple 0-shot user instruction. However, in [Ubani et al. \(2023\)](#) besides the 0-S scheme, FS was also used as ICL technique to generate synthetic data samples. Is fact, as far as data generation is considered, this was the only paper which used FS for the prompt. Other works, while keeping a 0-S user instruction scheme, the usage of the system instruction was leveraged, specifically in the works developed in [Møller et al. \(2024\)](#) and [Dai et al. \(2023\)](#). For more details on the prompts used, check the original papers. Besides the prompt, in [Møller et al. \(2024\)](#) was mentioned another relevant component, which is the LLM temperature. In this work, the temperatures of GPT-4 and Llama were set to 1 as that way the LLMs would generate samples with a larger diversity.

As for the usage of LLMs as text annotators, a wider variety of prompt schemes was used relative to their usage for data generation. Starting with the papers which used LLMs as baselines, in [Tang et al. \(2023\)](#) and [Møller et al. \(2024\)](#) ChatGPT, GPT-4, and Llama-2 were used in a 0-S scheme. As aforementioned, in [Ubani et al. \(2023\)](#) no LLM was used as baseline performance. As for the works which used LLMs as classifiers against crowdsourced annotators, in [Gilardi et al. \(2023\)](#), although it was not perfectly clear, it appears that a 0-S scheme together with system instructions for each classification task employed was used with GPT-3.5. Also in this work, two temperature values were used: 1 and 0.2, the latter so GPT-3.5 returns more deterministic outputs considering a lower temperature reduces the token sample to which nucleus sampling is applied.

However, the prompt used in [He et al. \(2024\)](#) for inference was quite different from all the previously seen prompts. Such paper, as far as I know, is the only paper which combined a FS prompt with CoT for text generation, FS-CoT. As explained in the Background, more specifically in Section 2.2.2.2, such technique is mainly applied to tasks that require analytical or logical reasoning, for example. In [He et al. \(2024\)](#), however, the authors were able to apply such concept to generate annotations. To do so, each FS example was given individually to ChatGPT and the LLM was asked to explain the annotation made in such example. After having the explanation for each FS example, a 4-S and 8-S prompt was built with CoT explanations.

### 3.2.3 Privacy Disclosure and Realism

BLEU [Papineni et al. \(2002\)](#) and ROUGE [Lin \(2004\)](#) are metrics that have been used frequently when it comes to quantify privacy disclosures in the SDG process in healthcare. In the case of BLEU scores, for example, lower scores mean a lower overlap between n-grams in the real texts relative to synthetic ones. This was done in [Ive et al. \(2020\)](#) [Guan et al. \(2021\)](#) [Libbi et al.](#)

(2021) Begoli et al. (2018). However, from all of those works, Ive et al. (2020) was the one that showed a more complete work in privacy disclosure by considering not only other metrics, like perplexity Jelinek et al. (1977), TER Snover et al. (2006), but also employed an outlier approach. Samples with n-grams in the lower quartile of frequency are used to generate new EHR's, and then the percentage change of rare n-grams from the input sequences to the synthetic sequences is measured. The results showed a rare n-gram presence increase of 8% in the SD, showing that the model memorised 8% of the rare n-grams. Libbi et al. (2021), on the other hand, examined privacy concerns and surveyed the top 122 synthetic EHRs with the highest ROUGE-N recall score, concluding that the most significant privacy concern occurred when whole chunks of text were copied during the generation of rare events.

As for realism, fewer works have delved into this topic. Regarding resemblance, any of the works with a granularity of text for EHR did anything regarding that. Only Yoon et al. (2023) did such task by implementing statistical similarity between the features of the real and generated EHRs. Utility is the branch of realism that has seen more work, however still falls short when compared to the interest given by research in dealing with privacy disclosure. Only two works with text EHRs measured utility, which were Guan et al. (2021) Libbi et al. (2021). Both of these use as a basis for comparison the results of TRTR, however while Guan et al. (2021) compared those results with TSTR, Libbi et al. (2021) instead compared with TRSTR.

Although Li et al. (2023) noticed a trade-off between privacy disclosure and utility with time series EHRs, that is not something that was mention in textual EHRs.

Other works didn't consider privacy disclosure and realism at all, specifically Lu et al. (2021) Frei and Kramer (2023) Lee (2018).

**LLM Applications** As far as privacy disclosure is concerned, none of the works developed using LLMs considered it. This is something that differs from the works developed with non-LLM applications.

However, when it comes to realism, and more specifically resemblance, here is where a new perspective was found in the literature on how to go about it with SD at the text granularity. For such topic of resemblance, nothing was made in He et al. (2024) and Gilardi et al. (2023) considering the focus on generating annotations and not synthetic samples. As for the remaining 4 works, only in Møller et al. (2024) did not do anything towards resemblance.

In Tang et al. (2023), Ubani et al. (2023), and Dai et al. (2023) sentence embeddings were used, however while in Ubani et al. (2023) and Dai et al. (2023) the similarity between those embeddings was measured using the cosine similarity, in Tang et al. (2023) T-SNE, which in principle is very similar to PCA, was used to map such sentence embeddings into a 2-dimensional latent space. The usage of T-SNE enabled the authors to identify that synthetic sentences showed a quite different distribution relative to the real sentence's distribution on the latent space and thus raising a possible correlation between such distribution entropy and performance differences. Besides sentence embeddings, in Ubani et al. (2023) cosine similarity was also applied to TF-IDF features.

As for the utility component of realism, every paper measured it. While in the background it was mentioned several utility schemes, the ones found in these LLM applications were TSTR, TRTR (baseline performance), and TRSTR. The review of utility will start with the works that focused on SDG, specifically [Tang et al. \(2023\)](#), [Ubani et al. \(2023\)](#), [Møller et al. \(2024\)](#), and [Dai et al. \(2023\)](#), and then jump to those which focused on comparing the LLMs inference performance against crowdsourced human resources, in [He et al. \(2024\)](#) and [Gilardi et al. \(2023\)](#). The TSTR utility scheme was deployed in [Tang et al. \(2023\)](#) and [Ubani et al. \(2023\)](#). As for the utility scheme with an augmented training dataset, thus a TRSTR scheme, it was considered in [Ubani et al. \(2023\)](#), [Møller et al. \(2024\)](#) and [Dai et al. \(2023\)](#).

When it comes to the works developed in [He et al. \(2024\)](#) and [Gilardi et al. \(2023\)](#), the utility was measured differently. In [He et al. \(2024\)](#), gold standard annotations were developed, however then crowd annotators and LLMs, in this case, GPT-4 and Llama-2, would annotate the same texts and be evaluated against the ground truth (gold standards). In [Gilardi et al. \(2023\)](#), the basic concept is the same, however in this case the tweets and news datasets already had the ground truth. Therefore, no model training was applied, but rather ICL using prompt engineering techniques, which were described in Section 3.2.2. Hence, it can be said that the utility scheme used was TRTR, however, the LLMs were not baselines in the particular main concept of these two works, which is annotation generation against crowd resources. In both these works the classifier was not trained with fine-tuning, but rather with ICL.

Following up on the contrast between the utility approach developed in [He et al. \(2024\)](#) and [Gilardi et al. \(2023\)](#), in 3 of 4 of the other works, specifically in [Tang et al. \(2023\)](#), [Møller et al. \(2024\)](#), and [Dai et al. \(2023\)](#), LLMs were also used as annotators, but because the particular main concept was fine-tuning a classifier, such TRTR scheme with ICL was considered a baseline performance.

### 3.3 Narrative Extraction

In this section of the Literature Review, works on extracting narratives, mainly the extraction of participants, will be explored. Several factors were considered in the search for such papers, mainly the search for the most recent works related to the healthcare sector. Not only that, but also works that used SD plus all the factors mentioned. Developments in Portuguese are less present than in English, so criteria such as using SD for participant extraction were much more difficult.

This section considers 18 papers for NER, which are repeatedly described in each subsection of Section 3.3.2. Not all papers are necessarily mentioned on all sections, mainly considering possible lack of information. Thus, this section focuses on one topic at a time while discussing all the works considered for this literature review. As for the SDG component of the literature review, the paper selection criteria included the SJR (Scimago Journal Ranking) score (when available), the number of citations, and the number of influential citations. Consensus is a great online tool that shows these metrics. Three of them are archive papers, but besides that all of them are published

in Q1 journals or conferences. Some papers have very few citations, generally in a few recent works.

This literature review section starts with Section 3.3.1 describing the preprocessing and parsing steps made in the literature. Section 3.3.2 details the extraction of participants, those being LMs used as text encoders and model architectures for NER, datasets used, and evaluation metrics.

Because the Portuguese references for healthcare were scarce, this search was expanded to a more general context of NER. The following references are the only ones for NER that aren't from the healthcare domain: Souza et al. (2019), Santos et al. (2019), and Quinta de Castro et al. (2018).

### 3.3.1 Pre-processing and Parsing

The immediate trend in the literature is the impact of the LM architecture on text preprocessing and parsing. Architectures that don't use BERT-based LMs to compute embeddings have a larger pipeline with more features extracted from text. On attention-based text encoders, such as BERT, the LM mainly receives as input the tokens created by the respective BPE tokenizer. The other features such LMs can receive are the attention masks and a binary feature that shows the LM which tokens are from which sentence. For further details on such LMs, particularly BERT-based, details are shown in Section 2.3.3.2

In Solarte-Pabón et al. (2021), Tu et al. (2023), Lopes et al. (2019), Ruan and Lee (2018), de Souza et al. (2019), and Lopes et al. (2020) POS are extracted. Within these, in Solarte-Pabón et al. (2021), Lopes et al. (2019), and Lopes et al. (2020) lemmatization is applied, using the lemmas as additional features. Solarte-Pabón et al. (2021), Lopes et al. (2019) and Lopes et al. (2020) are papers for Portuguese, hence they will be further detailed below. In Ruan and Lee (2018), the POS tagger used is the Stanford POS Tagger. Besides POST and lemmatization, other features are used in the literature. Besides POST, in Tu et al. (2023) parsing is also applied using tools to do so, in this case, GATE, cTAKES, and MedSpaCy. In Ruan and Lee (2018), semantic features are also extracted, while in Lopes et al. (2019), de Souza et al. (2019), and Lopes et al. (2020) morphological and orthographic features are extracted.

Another preprocessing procedure that is used is sentence segmentation. Here, there are two main approaches. The first is to use a sentence tokenizer, for example, in Tu et al. (2023). The second approach is the usage of a pre-defined stride, where the goal is not to retrieve sentences but chunks of text, which was applied, for example, in Souza et al. (2019). Case folding is another preprocessing technique, which was used, for example, in Schneider et al. (2020), Tseo et al. (2020), and Quinta de Castro et al. (2018).

Now focusing on the preprocessing and parsing tools available, spaCy Honnibal and Montani (2017) is an NLP tool that can accomplish several tasks, such as sentence segmentation, tokenization, lemmatization, and POST. There are other tools, such as NLTK Bird and Loper (2004), medspaCy Eyre et al. (2021), and scispaCy Neumann et al. (2019). In the case of medspaCy, it can also do the preprocessing tasks spaCy does. In the case of the sentence tokenizer, for example, it was used a rule-based algorithm specifically developed for clinical texts, RuSH Shi et al. (2016).

**Developments in Portuguese** The trend mentioned above can also be seen in Portuguese applications. The papers [Lopes et al. \(2019\)](#) and [Lopes et al. \(2020\)](#) come from the same authors, where tokenization, POST, and lemmatization are applied using a tool called NLPPort [Rodrigues et al. \(2018\)](#). A more updated version was released by two of the three authors that developed NLPPort, NLPyPort [Ferreira et al. \(2019\)](#), which uses NLTK as a basis to develop a better-suited tool for tokenization, POST, and lemmatization for Portuguese NLP use cases.

spaCy [Honnibal and Montani \(2017\)](#) can also perform well in these tasks in Portuguese. In addition to tokenization, POST, and lemmatization, spaCy for Portuguese has other models to perform other preprocessing tasks, such as sentence segmentation, extracting morphological features, and a dependency parser. Such models are not rule-based but rather based on ML; more specifically, the models are neural-based architectures<sup>2</sup>. For example, in Section 2.3.1.1 it was shown a neural-based architecture for sentence segmentation.

### 3.3.2 Identification and Extraction of Components

The aforementioned papers regarding the participant extraction component will be explored in this section. In this regard, the literature review is further segregated into several other sections.

Section 3.3.2.1 deals with the LM and classifier architectures used for NER. Section 3.3.2.2 shows the details of the datasets used to train the LMs (when applicable) and the classifiers. Section 3.3.3 details the specifics of the evaluation scores or datasets used to assess the NER classifiers.

#### 3.3.2.1 LMs and classifier architectures

As in Section 3.3.1 where, the features were either extracted or not depending on the LM architecture, also, here, the literature can be divided into two main groups the same way: papers that use WE-based LMs and BERT-based LMs.

Not all papers go into much detail on the WEs themselves. In [Tseo et al. \(2020\)](#), [Tu et al. \(2023\)](#), [Ullah Miah et al. \(2023\)](#), and [Solarte-Pabón et al. \(2021\)](#), in-domain embeddings are used. The work developed in [Tseo et al. \(2020\)](#) is one of the few that develops more LM details. They used the CBOW model with the FastText algorithm to train WEs. This is the only paper discriminating between CBOW and Skip-Gram regarding which word2vec model was used. [Tu et al. \(2023\)](#) also used word2vec. However, they also used gloVe and one-hot embeddings, achieving the highest performance with word2vec. The FastText algorithm was also used to train general-domain word2vec embeddings in [Solarte-Pabón et al. \(2021\)](#). In [Ruan and Lee \(2018\)](#), which used a simple CRF model, no LM was mentioned.

As for BERT-based LMs, two papers employed them. In [Liu et al. \(2021\)](#), besides the usage of the original BERT, the authors also used BioBERT ([Lee et al. \(2019\)](#)), ClinicalBERT ([Huang et al. \(2019\)](#)), and BlueBERT ([Peng et al. \(2019\)](#)). In [Lee et al. \(2019\)](#), the original BERT was fine-tuned on biomedical domain corpora (PubMed abstracts and PMC full-text articles), thus creating the

---

<sup>2</sup><https://github.com/explosion/spaCy>

pre-trained LM named BioBERT. This LM outperformed several SOTA models on several MLP downstream tasks, such as NER and RE. In [Huang et al. \(2019\)](#), ClinicalBERT was pre-trained on a large corpus of clinical texts, being able to outperform BERT on MLM and NSP on clinical texts significantly. In [Peng et al. \(2019\)](#), BlueBERT was pre-trained by fine-tuning BERT-base and BERT-large on the MIMIC-III corpora and PubMed abstracts. BlueBERT achieved SOTA performance on several datasets, even surpassing BioBERT.

When using WEs as the LM, the literature showcases an abundant preference for the usage of LSTM cells, more specifically for a BiLSTM component, which encodes long-term dependencies in both directions. Such component can be seen being used in [Tseo et al. \(2020\)](#), [Tu et al. \(2023\)](#), [Ullah Miah et al. \(2023\)](#), and [Solarte-Pabón et al. \(2021\)](#). As mentioned above, in [Ruan and Lee \(2018\)](#), only a CRF model is used. Despite that, such classifier architectures use other sub-layers than a BiLSTM one. In [Tseo et al. \(2020\)](#) and [Solarte-Pabón et al. \(2021\)](#), a CRF sub-layer is stacked on top of the BiLSTM. Such a sub-layer is important to receive the token-level predictions given by the BiLSTM sub-layer and thus consider the inherent label sequence dependency that exists in NER. Both papers employed the IOB tagging scheme, and as such, the B and I prefixes follow a sequential rule, which a BiLSTM sub-layer can not tackle, but a CRF sub-layer can.

**Developments in Portuguese** In contrast with the literature review detailed above, the WE-based papers provide more details about the models and algorithms used for Portuguese. In [Lopes et al. \(2019\)](#) and [Lopes et al. \(2020\)](#), in-domain WEs were trained with a Skip-gram model using the FastText algorithm. [Quinta de Castro et al. \(2018\)](#) also applied the skip-gram model to compute WEs, however besides FastText, GloVe and word2vec were also used. The other paper that did not use a BERT-based LM, which is [Santos et al. \(2019\)](#), not only applied both word2vec and FastText algorithms but also both WE models were also used, CBOW and Skip-gram.

Regarding BERT-based LMs, in [Schneider et al. \(2020\)](#), 3 PLM were trained using transfer learning from BERT-Multilingual-Cased, a PLM model trained in [Devlin et al. \(2019\)](#) composed of a training composed of texts on several languages, including Portuguese. The three PLM instances fine-tuned were BioBERTpt(bio), BioBERTpt(clin), and BioBERTpt(all). The training corpus comprised biomedical literature, clinical notes from Brazilian hospitals, and both. Such dataset is called SemClinBr ([Oliveira et al. \(2022\)](#)). For more details on this dataset, see Section 3.3.2.2. In [Souza et al. \(2019\)](#), two BERT LMs, PT-BERT Base and PT-BERT Large, were pre-trained, being initialized using the BERT-Multilingual-Cased Base and (English) BERT-Cased Large, respectively. Check the original work for more details on both pre-training procedures and hyperparameters for these two papers.

Although there is no reference yet, a new family of PLMs was made available for European Portuguese. However, instead of being BERT-based, these have as their backbone a DeBERTa model. This family of LMs is called MediAlbertina<sup>3</sup>. These models made further MLM pre-training based on the Albertina models, developed in [Rodrigues et al. \(2023\)](#). These models

---

<sup>3</sup><https://huggingface.co/portugueseNLP>

were trained on electronic medical reports shared by the largest public hospital in Portugal. Two versions were made available, one with 900M and another with 1.5B parameters.

As for the classifier architectures, in [Lopes et al. \(2019\)](#), [Souza et al. \(2019\)](#), [Santos et al. \(2019\)](#), [Quinta de Castro et al. \(2018\)](#), and [Lopes et al. \(2020\)](#) a CRF sub-layer is placed at the end of the classifier architecture to consider the token-level tag inherent sequence for the tagging schemes, which was either IOB or IOBES. Something that was not seen in the non-Portuguese component of this literature review was the usage of a CRF sub-layer on top of a classifier using a BERT-based LM, however this was the case in [Souza et al. \(2019\)](#). The papers that used a WE-based LM for their classifiers also used a BiLSTM sub-layer before the CRF sub-layer, those being [Lopes et al. \(2019\)](#), [Santos et al. \(2019\)](#), [Quinta de Castro et al. \(2018\)](#), and [Lopes et al. \(2020\)](#). An interesting architecture was used in [Lopes et al. \(2020\)](#), where residual connections are made between the LM and feature embeddings and each sub-layer, also considering each cell of the BiLSTM sub-layer. This means such input is given to forward and backward LSTM cells and the CRF sub-layer.

Like in the non-Portuguese works, in [Schneider et al. \(2020\)](#), the classifier architecture comprises the LM and an output layer with three output perceptrons, one for each tag (O, B, or I). This, as mentioned above, contrasts the classifier architecture used in [Souza et al. \(2019\)](#), where the CRF sub-layer was placed on top of the output layer. In [Souza et al. \(2019\)](#), the PT-BERT-CRF architecture was used in two settings, one in which PT-BERT had its trainable parameters frozen and the other, with the latter setting outperforming the former. This was made to check if the sub-layers on top of the LM could learn relevant features for the NER task, thus being called by the authors a feature-based approach.

### 3.3.2.2 Datasets

The datasets used can be divided into two categories: LM datasets and NER datasets. LM datasets are used to pre-train an LM, while a NER dataset is used to train a classifier that uses the knowledge gathered by the LM from the LM dataset corpus. In the literature, it is often made a separation between the training NER and the testing NER datasets because, in several cases, they are not the same, such as in [Lopes et al. \(2019\)](#), [Schneider et al. \(2020\)](#), and [Lopes et al. \(2020\)](#). Here, however, the focus is on the dataset availability. Hence, such distinction was not a concern.

The work developed in [Liu et al. \(2021\)](#) and [Tseo et al. \(2020\)](#) used the same NER dataset, which consists of 3,314 clinical trials, with 121,221 clinical entities, attributes, and limits, randomly sampled from ClinicalTrials.gov. Check the original work developed in [Tseo et al. \(2020\)](#) for further participant details. On the other hand, in [Tu et al. \(2023\)](#) and [Ruan and Lee \(2018\)](#), the i2b2 challenge NER dataset is used, where the former uses the 2009 and 2012 tracks, while the latter uses just the 2009 track with the medication corpus.

In [Ullah Miah et al. \(2023\)](#), 3,145 sentences were annotated from 50 articles selected by an expert domain. Such annotations were made for two participants, drugs and diseases. In [Solarte-Pabón et al. \(2021\)](#), 400 documents of patients with lung cancer from the Hospital Universitario Puerta de Hierro, Madrid, Spain, were used and separated into 14,500 sentences. On the other

hand, this dataset has annotations for many more participants than the one from Ullah Miah et al. (2023). In this dataset, one of the NER tags is called Event, which, although it can be recognized by a classifier in the same participant can, is still a different concept relative to a participant. Further details on such distinction can be seen in Sections 2.3.2.1 and 2.3.2.3 for participants and events, respectively. In Chuang et al. (2023), 8,125 clinical notes were retrieved within one month of examination from a total of 5,495 patients.

**Developments in Portuguese** Contrary to the literature review above, which didn't mention specifically which LM datasets were used to train the LMs, like the biomedical pre-trained embeddings mentioned in Solarte-Pabón et al. (2021), in Portuguese, this was given more emphasis.

To develop the works in Lopes et al. (2019) and Lopes et al. (2020), in-domain WE were trained with the FastText algorithm using 3 377 clinical texts, which were retrieved from all volumes of the Synapse journal, from 2001 to 2018. In Schneider et al. (2020), BioBERTpt was fine-tuned from a previous pre-trained BERT model, mentioned in Section 3.3.2.1, using 2 different training corpus. One consisted of a corpus of 2,100,546 clinical notes with 3.8M sentences and 27.7M words. The second LM dataset consisted of titles and abstracts from Portuguese scientific papers published in PubMed and Scielo, with 737,469 sentences and with around 16.4M words. As for the clinical notes, these EHRs came from several hospitals in Brazil.

As for LM datasets not in the healthcare domain, both Souza et al. (2019) and Santos et al. (2019) used the brWaC dataset, however Santos et al. (2019) also used BlogSet-BR and ptwiki-20190301.

As for NER datasets, in Lopes et al. (2019), Schneider et al. (2020), and Lopes et al. (2020), it was used a dataset of 281 clinical texts from the numbers 1 and 2 of volume 17 published by the Portuguese Society of Neurology in the Synapse journal. It is called PortugueseClinicalNER considering its GitHub page<sup>4</sup>. Such dataset contains participants annotated for the following tags: Characterization, Test, Evolution, Genetics, Anatomical Site, Negation, Additional Observations, Condition, Results, DateTime, Therapeutics, Value, and Route of Administration. Such participant types are detailed in Lopes et al. (2019). A much smaller NER dataset was used in two of these three works, specifically in Lopes et al. (2019) and Lopes et al. (2020). Such dataset contains 20 clinical texts from the Neurology service of Coimbra University Hospital Centre (CHUC), in Coimbra, Portugal.

Another NER dataset used is called SemClinBr Oliveira et al. (2022), used in Schneider et al. (2020). This dataset comprises 1,000 EHRs with 65,129 participants and 11,263 relations. Regarding the participants, these were annotated using 100 UMLS semantic types (STY) which were aggregated into 13 participant types: Disorders, Chemicals and Drugs, Medical Procedure, Diagnostic Procedure, Disease Or Syndrome, Findings, Health Care Activity, Laboratory or Test Result, Medical Device, Pharmacologic Substance, Quantitative Concept, Sign or Symptom and Therapeutic or Preventive Procedure. The annotation process had two main phases. First, UMLS was used to annotate participants automatically, and then a more curated approach with in-domain

<sup>4</sup><https://github.com/fabioacl/PortugueseClinicalNER>

experts to improve the annotations. Such annotation procedure is very similar to INCEpTION, an annotation tool developed in [Klie et al. \(2018\)](#).

As for the remaining papers which are not from the healthcare domain, that being [Souza et al. \(2019\)](#), [Quinta de Castro et al. \(2018\)](#), and [Santos et al. \(2019\)](#), the NER datasets used were the HAREM I and Mini-HAREM for training and testing, respectively.

### 3.3.3 Evaluation

In this section, a review on the literature of evaluation scores is made. As previously mentioned, the main downstream NLP task considered for this dissertation is participant extraction. Hence, the literature review made in this regard is focused on the evaluation of NER systems.

As made in Sections 3.3.1 and 3.3.2, the review on evaluation scores is made firstly for non-Portuguese works, and then a separated part just for the Portuguese-related papers. This was done given the particular emphasis of our work in the Portuguese language.

#### 3.3.3.1 Evaluation scores

Overall, in the literature, there are two main ways of evaluation how a classifier extracts participants from clinical or biomedical texts: using more strict scores or more relaxed scores. In some papers, however, it was quite difficult to find what scores were used. The metrics themselves are always the same: precision, recall, and f1-score, although in some papers only the f1-score is shown.

Evaluation using exact entity type and partial boundaries was used in [Liu et al. \(2021\)](#), [Tseo et al. \(2020\)](#), and [Ullah Miah et al. \(2023\)](#), while more strict scores are used in [Solarte-Pabón et al. \(2021\)](#), [Gao et al. \(2021\)](#), and [Zhang et al. \(2021\)](#). At the token level, in [Ruan and Lee \(2018\)](#) and [Tu et al. \(2023\)](#) and [Kocaman and Talby \(2021\)](#), the metrics are not measured considering any strict or partial score, but rather each individual tag is evaluated, meaning B and I tags for each participant type. Despite in [Ruan and Lee \(2018\)](#) and [Tu et al. \(2023\)](#) the metrics results were shown at the B and I tag levels, in [Kocaman and Talby \(2021\)](#), however, just the micro-average results were shown. In [Hakala and Pyysalo \(2019\)](#), both exact and partial boundary matching was used.

A difficulty found when reviewing a few papers was the way the metrics were calculated in terms of TP, FP, and FN, which was either not detailed or referenced. This happened in [Liu et al. \(2021\)](#), [Chuang et al. \(2023\)](#), and [Tseo et al. \(2020\)](#). In [Solarte-Pabón et al. \(2021\)](#), [Gao et al. \(2021\)](#), and [Zhang et al. \(2021\)](#) the metrics calculation details are shown. In fact, the paper with the evaluation procedure more robust did not require to add such equations to the paper itself, such work being made in [Gao et al. \(2021\)](#). From all non-Portuguese papers reviewed, this was the only one that used an evaluation script benchmark, SemEval 2013 Task 9.1 evaluation script ([Segura-Bedmar et al. \(2013\)](#)). In [Hakala and Pyysalo \(2019\)](#), although no reference or equation was shown, the explanation was clear enough to understand the details of the evaluation scores used.

A different partial score was developed in Ullah Miah et al. (2023) and Hakala and Pyysalo (2019). In Ullah Miah et al. (2023), a correct prediction at the entity level would be considered only if the first token of such prediction matched the ground-truth. In Hakala and Pyysalo (2019), on the other hand, the evaluation concept implemented in Ullah Miah et al. (2023) was generalized to consider also the last token being required to also be the last in the ground-truth. As mentioned above, exact boundary and any overlap boundary matching was implemented.

**Developments in Portuguese** As for works developed in Portuguese, in contrast to the non-Portuguese papers, the explanation of the evaluation scores was much clear, as well as the usage of evaluation benchmark.

In Lopes et al. (2019) and Lopes et al. (2020) used strict and relaxed boundary scores, although no evaluation script was mentioned. As with non-Portuguese works, the usage of metrics at the token-level, meaning results for B and I tags, was also reported, namely in de Souza et al. (2019).

As for the usage of evaluation scripts, in contrast with non-Portuguese paper that used the SemEval 2013 Task 9.1 as evaluation script, work developed in Gao et al. (2021), in Portuguese-related works two CoNLL evaluations scripts were used, the CoNLL-2002 and CoNLL-2003 evaluation scripts. The former was used in Santos et al. (2019) and Quinta de Castro et al. (2018), while the latter was utilized in Souza et al. (2019).

In Schneider et al. (2020), a micro average is used where the results are computed over all participant instances and then over all participant types. That being said, no information about the score was found, meaning either exact or partial boundaries matching.

### 3.4 Summary

This chapter aimed at discussing and studying the current state of the art regarding both SDG and narrative extraction, mainly identification and extraction of participants. RE was also considered given how the work developed in this dissertation might be used for linking components, more specifically that NLP downstream task.

The literature review discussed recent developments in generative model architectures and emphasized LLM applications, particularly in the healthcare domain and in Portuguese. It highlighted the shift from traditional word embedding models to transformer-based text encoders, such as BERT-based models, in sequence tagging model architectures applied to NER.

## Chapter 4

# Work Methods

This chapter will focus on the work methods used for this dissertation, including generating SEHRs, annotating them automatically, and extracting participants. It builds on the Literature Review in Chapter 3. The results achieved with the work methods used can be seen in Chapter 5.

### 4.1 Outline

Section 4.2 will cover the generation and annotation of SEHRs. Section 4.2.1 details generating SEHRs using GPT-3.5. Section 4.2.2 outlines annotating SEHRs using a tool we developed that relies on UMLS. Section 4.2.3 explores generating and annotating SEHRs with GPT-4o.

Section 4.3 will cover all the work methods for participant extraction, NER. Section 4.3.2 will describe the central experiment scheme and the various experiments. Section 4.3.3 will explain the preprocessing details. Section 4.3.4 will provide an in-depth look into the training pipeline, and Section 4.3.5 will mention the metrics used to evaluate the NER models.

### 4.2 SEHRs document generation and annotation

This section describes all the details regarding the work methods employed. Those details consider the initial approach to generating SEHRs with GPT-3.5 and the several prompts developed. Those SEHRs were then annotated with a weak annotation tool we developed using UMLS. Throughout this work, will refer to the UMLS Metathesaurus as just UMLS. More specifically, the biomedical vocabulary in Portuguese.

Besides the details of that initial approach, this section also contains details on our usage of GPT-4o to generate and annotate SEHRs. Several experiments were made, thus making Section 4.2.3 relatively complex and harder to understand.

Figure 4.1 shows of an overview of the workflow used to generate and annotate SEHR. It's important to note that ChatGPT refers to the use of models like GPT-4o or GPT-3.5 in the chat

interface provided in the OpenAI’s website<sup>1</sup>, while the use of the OpenAI API<sup>2</sup> refers to its usage in Python. When specific model names, such as gpt-3.5-turbo or gpt-4o-2024-05-13, are mentioned, it implies the usage of the OpenAI API. This distinction is crucial as both ChatGPT and the OpenAI API are utilized in various parts of this work.

An easier way to distinguish is the following. When a small sample of SEHRs is generated and/or annotated, ChatGPT was used. Otherwise, the OpenAI API was used. For example, all GPT-4o experiments until the generation of 2 datasets with 300 SEHRs were made with ChatGPT.

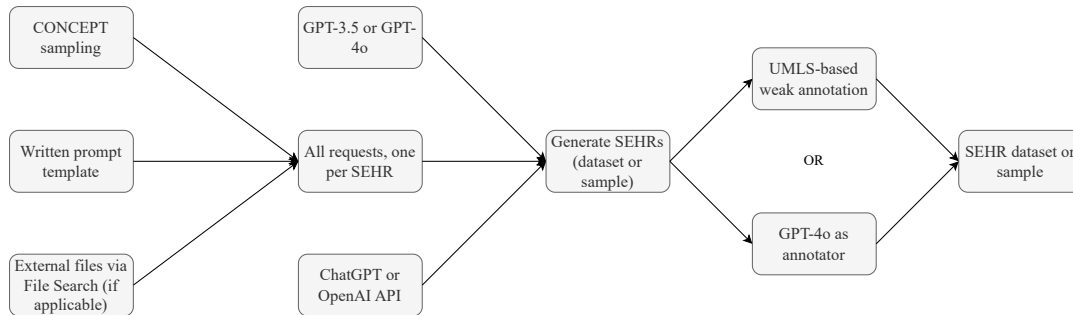


Figure 4.1: Diagram of the overall work method followed to generate and annotate SEHRs. CONCEPT is a disease or condition randomly sampled from UMLS to generate SEHRs for it. The SEHR dataset or sample output is the same component used as input in Figure 4.2

#### 4.2.1 GPT-3.5

We conducted initial prompt engineering tests to generate SEHRs using GPT-3.5 for this dissertation project. We adjusted the prompt template to improve the quality of the SEHRs, which were evaluated qualitatively by an oncology expert. The details of that evaluation will be described below. The description of the several prompts developed is written below. These prompts were tested with ChatGPT. The actual prompts can be seen in Section 5.2.1.1.

To generate more varied SEHRs, a disease or condition would be randomly sampled with replacement from a KB with medical concepts. We chose to use UMLS because of its wide variety of Portuguese concepts. We called that disease or condition component the CONCEPT. Certain concepts had definitions in UMLS.

- **Prompt 1** — The Prompt 1 template includes only the task description. Prompt 1 randomly selects a disease or condition from UMLS to instruct GPT-3.5 in creating a SEHR. Two versions of Prompt 1 were created: 1.1 and 1.2. Prompt 1.2 included the CONCEPT’s definition, while Prompt 1.1 did not;
- **Prompt 2** — We used FS-prompting as the basis for new prompt templates. Hospital P in northern Portugal provided de-identified real EHRs. Our goal was to prompt GPT-3.5

<sup>1</sup><https://chatgpt.com>

<sup>2</sup><https://openai.com/api/>

to generate SEHRs in European Portuguese. Prompt 1.1 was adapted into Prompt 2, while Prompt 1.2 was excluded;

- **Prompt 3** — While Prompt 2 was in a 1S format, with Prompt 3 we tested 2S and 4S prompting so GPT-3.5 had more external context;
- **Prompt 4** — Besides using FS examples, we added external knowledge from UMLS to the prompt with the CONCEPT. That knowledge consisted of definitions of the FS condition and CONCEPT. This was done in Prompt 1.2 too. However, now that knowledge was structured considering the FS template. First the task definition, then the external knowledge, and then the FS example;
- **Prompt 5** — Considering the healthcare expert's critics on the SEHRs generated with Prompt 3 and 4, the final prompt was achieved in a 2S format without external knowledge augmentation, with some changes to the prompt's text. These changes, such as explaining certain acronyms or abbreviations, will become evident in Section 5.2.1.

While developing the several prompts, we generated a few SEHRs for a single CONCEPT, Acute Myeloid Leukemia. This was done to, as mentioned before, to improve the prompts with the help of an expert in oncology. The SEHRs needed to resemble real medical records regarding their structure, misspellings, and summarizing level. The SEHRs were required to include specific medical data, and be written in European Portuguese.

The FS examples from Hospital P used in prompts 2 to 5 used as CONCEPT pneumonia and acute decompensated heart failure. Using Prompt 5, we created 1,200 SEHRs using gpt-3.5-turbo. We also utilized gpt-3.5-turbo to generate a second synthetic corpus using Prompt 5. We replaced the 2S examples with 2 EHRs from SemClinBr (Oliveira et al. (2022)). The SEHRs generated with 2S examples from Hospital P are called pSEHRs, and the ones that use 2S examples from SemClinBr are called bSEHRs, where the "b" stands for benchmark. The bSEHRs generated had more Brazilian Portuguese text given the benchmark's EHRs language.

#### 4.2.2 UMLS-based weak annotation

An LLM such as GPT-3.5 can generate and annotate SEHRs. The literature review has highlighted that using an LLM annotator is possible. However, we did not use GPT-3.5 as an annotator for three main reasons.

Firstly, GPT-3.5 is constrained to 4,096 tokens per prompt-output pair, and Prompt 5 already contains almost 3,000 tokens. Instructing GPT-3.5 in generating and annotating SEHRs would result in a much smaller SEHR with fewer annotations, possibly with lower quality.

Secondly, the UMLS in Portuguese encompasses thousands of concepts appropriately labeled with their semantic types (STY), making it a robust KB of healthcare-related concepts. UMLS was also utilized as a foundation for annotating the SemClinBr benchmark.

Lastly, a key component of our research plan involved developing a UMLS-based tool that matched text spans in EHRs with UMLS concepts, thus annotating them. We called it UMLS-based weak annotation tool. This is explained next.

We used this tool to annotate the pSEHR dataset and the bSEHRs generated. More details on their generation are explored in Section 4.2.1.

#### 4.2.2.1 UMLS data pre-processing

After developing the initial pipeline to generate SEHRs with GPT-3.5, explained in Section 4.2.1, the next step was to create the UMLS-based weak annotation tool. We noticed that accessing and retrieving UMLS data from local files was more time-efficient than using an API key. We retrieved all the UMLS data in Portuguese from its 2023AA version.

The second step was to filter the vast amount of data retrieved from UMLS. We only considered the UMLS Portuguese concepts with the final set of STYs chosen for the downstream NER task. Those STYs are mentioned in Section 4.3.2.1. The preprocessed data was stored in a data structure where each row consisted of a concept with three columns: a unique identifier (CUI), the concept's text (STR), and the respective STY.

#### 4.2.2.2 Weak annotation pipeline

After obtaining the preprocessed UMLS data, we developed the UMLS-based weak annotation tool to identify exact string matches between text spans in SEHRs and UMLS concepts, STRs. We extended the algorithm to work for n-grams up to  $n = 5$ , as this was the highest number of words in any STR found in the preprocessed UMLS dataframe.

Table 4.1 shows all n-grams up to  $n = 3$  present in the sentence "Relata alucinações somáticas frequentes, onde sente que insetos estão a rasteja sob a sua pele". The n-grams retrieved from the pSEHRs and bSEHRs were sorted from 5-grams to unigrams and stored in a dictionary. The respective start and end relative positions were also stored in the dictionary for the respective n-gram.

It's important to note that simply computing all n-grams is not enough for accurately annotating the SEHRs. For example, a trigram consists of 2 bigrams and 3 unigrams, or the trigram's sub-n-grams. If a trigram has an exact match in UMLS, its sub-n-grams should not be considered for annotation in subsequent iterations. These sub-n-grams are excluded based on their relative positions within the trigram. For instance, if the same trigram has a span of (361, 402), the UMLS-based weak annotation tool removes the bigrams and unigrams that fall within this span. Thus, the dictionaries must be sorted from 5-grams to unigrams, as mentioned above. Table 4.1 has another example in its caption.

| 1-gram      | 2-gram                | 3-gram                           |
|-------------|-----------------------|----------------------------------|
| Relata      | Relata alucinações    | Relata alucinações somáticas     |
| alucinações | alucinações somáticas | alucinações somáticas frequentes |
| somáticas   | somáticas frequentes  | somáticas frequentes onde        |
| frequentes  | frequentes onde       | frequentes onde sente            |
| onde        | onde sente            | onde sente que                   |
| sente       | sente que             | sente que insetos                |
| que         | que insetos           | que insetos estão                |
| insetos     | insetos estão         | insetos estão a                  |
| estão       | estão a               | estão a rasteja                  |
| a           | a rasteja             | a rasteja sobre                  |
| rasteja     | rasteja sobre         | rasteja sobre a                  |
| sobre       | sobre a               | sobre a sua                      |
| a           | a sua                 | a sua pele                       |
| sua         | sua pele              |                                  |
| pele        |                       |                                  |

Table 4.1: N-grams (up to trigrams) present in the sentence "Relata alucinações somáticas frequentes, onde sente que insetos estão a rasteja sob a sua pele.". The trigram "alucinações somáticas frequentes" has the following sub-n-grams: "alucinações somáticas", "somáticas frequentes", "alucinações", "somáticas", and "frequentes". If the trigram has an exact match in UMLS, for example a symptom, then those other sub-n-grams won't be considered for further annotation.

Another component of the weak annotation tool is the TAG mappings. TAG mappings was an additional component considered to prepare the BIO tagging scheme. That being said, it does not affect the functioning of the algorithm.

Besides the dictionary of n-grams, the preprocessed UMLS dataframe and the TAG mappings, acronyms were also taken into consideration. When an n-gram consisted of an acronym (acronym table developed by experts), usually an unigram, the n-gram's search term would be replaced by its full text to go through UMLS. For example, the search term of the n-gram "LMA" would be "Leucemia mieloide Aguda". However, the annotated entity would still be "LMA". Besides this scenario, the search term in the n-gram. This was done because UMLS did not contain several acronyms we tested.

A default annotation is given to every n-gram, only changing when an exact match is found in the preprocessed UMLS dataframe. Such default annotation includes the CUI, STR, STY, and TAG. The default TAG is "O" when no match is found because that is the NER tag given to a text span with no participants. The UMLS-based weak annotation algorithm retrieves all matches found in UMLS, not just one or the first that appears.

**Algorithm 1** Weak Annotation with UMLS

---

**Input:** N-grams  $\leftarrow$  sorted array of {N-gram: N-gram text, Start: start position, End: end position} from one SEHR, UMLS preprocessed dataframe, acronyms, TAG mapping

**Output:** Results  $\leftarrow$  JSON, sorted by Start, of {N-gram: N-gram text, Result: annotations, Start: start position, End: end position} for a pSEHR

**Initialize:** Empty JSON file, Default annotation  $\leftarrow$  {CUI: "", STR: "", STY: "", TAG: 'O'}

**for** each N-gram in N-grams **do**

    Retrieve the search term, either N-gram or a mapping of an acronym

    Search and retrieve relevant records from UMLS preprocessed dataframe

**if** exact match found **then**

        Collect CUI, STR, and STY of the exact matches

        Retrieve TAG by mapping the STY to the value in TAG mapping

        Store annotations in JSON file

        Remove sub-n-grams from N-grams

**else**

        Assign the default annotation to N-gram without matches

**end if**

**end for**

---

The UMLS-based weak annotation tool developed can be seen in Algorithm 1. An example of an entity match, in this case an acronym, is shown below after applying Algorithm 1 to the unigram "RMN". The annotation is made to the text span that contains "RMN", but the search term is "Ressonância magnética nuclear".

```
{
  "ngram": "RMN",
  "result": [
    {
      "CUI": "C0024485",
      "STR": "Ressonância magnética nuclear",
      "STY": "Diagnostic Procedure",
      "TAG": "DiagnosticProcedure"
    }
  ],
  "start": 688,
  "end": 691
}
```

**4.2.2.3 CoNLL-2003 annotations**

After running Algorithm 1 on the n-grams of all SEHRs, we created files with CoNLL-2003 format to store the JSON annotations. We initially chose CoNLL-2003 format for SparkNLP<sup>3</sup>

---

<sup>3</sup><https://sparknlp.org>

compatibility, but later kept it when we chose to work with PyTorch. The conversion algorithm adds BIO prefixes to non-"O" TAGs by splitting the n-grams by words. The whitespace tokenizer is used to split by words.

A critical detail in this conversion algorithm pertains to handling sentences. In a CoNLL-2003 annotation file, sentences are separated by an empty line. The conversion algorithm uses a sentence tokenizer to identify sentence boundaries, and in cases where an entity's n-gram spans two sentences ( $n > 1$ ), the entity is moved to the following sentence to adhere to the BIO tagging scheme rules. In other words, because an input sequence can't start with half an entity (start with a label with prefix "I"), we do that. However, the sentence tokenizer does not work with n-grams, so the raw SEHRs generated are used to determine the sentence boundaries. The sentence boundaries are also spans, where the total number of spans equals the number of sentences. The sentence boundary detector is mentioned in Section 4.3.3.1.

To convert from JSON to CoNLL-2003 annotation format, Algorithm 2 was applied. An example of an annotation file with CoNLL-2003 format can be seen below.

---

**Algorithm 2** Convert Annotations to CoNLL-2003 Format with BIO tagging scheme

---

**Input:** annotations given by Algorithm 1, raw SEHRs  
**Output:** string with CoNLL-2003 annotation  
**Initialize:** list for CoNLL formatted data and include headers  
 Compute sentence boundaries using raw SEHRs  
**for** each annotation in annotations **do**  
   Extract TAG from the annotation's first result found  
   Extract the N-gram text and its End position  
   Split the N-gram by words (whitespace tokenizer)  
   **if** N-gram's End position is beyond current sentence boundary **then**  
     Add white space to the CoNLL-2003 annotation list to indicate a new sentence  
   **end if**  
   **for** each word in the N-gram **do**  
     **if** Tag is "O" **then**  
       Append word to the CoNLL-2003 annotation list  
     **else**  
       Determine tag prefix (B- or I-) and append to the CoNLL-2003 annotation list  
     **end if**  
   **end for**  
**end for**  
 Compile the CoNLL-2003 annotation list into a single string

---

```

Sem O O O
dor O O B-SignOrSymptom
de O O I-SignOrSymptom
ouvido O O I-SignOrSymptom
tonturas O O B-SignOrSymptom
ou O O O
zumbidos O O B-DiseaseOrSyndrome

Audiometria O O B-DiagnosticProcedure
tonal O O O
perda O O B-DiseaseOrSyndrome
auditiva O O I-DiseaseOrSyndrome
bilateral O O I-DiseaseOrSyndrome
simétrica O O O
de O O O
grau O O O
moderado O O O
a O O O
severo O O O

```

### 4.2.3 GPT-4o

This section will build on the work methods developed in Sections 4.2.1 and 4.2.2. The methodology for using GPT-4o is more extensive and detailed. Therefore, in this section, we have organized it into the several subsections used in Chapter 5.

Section 4.2.3.1 will discuss the methodology behind the SDG modifications made after the initial approach, as described in Sections 4.2.1 and 4.2.2. This section will explore a hypothesis we developed that led to significant shifts in this dissertation's work. We used a RAG tool to augment the written prompt.

Section 4.2.3.2 will continue from the methods explained in Section 4.2.3.1, detailing the adjustments made to the SDG pipeline to achieve its final and fully developed state.

Section 4.2.3.3 will cover the usage of the fully-developed SDG pipeline in a different and unbiased NER dataset, PortugueseClinicalNER (Lopes et al. (2019)). This required adjusting a few parameters of the SDG pipeline to accommodate new details, such as STYs to annotate and the file format for annotations.

#### 4.2.3.1 Comparative Study and Proof of Concept with an LLM as SEHR generator and annotator

Considering the initial results achieved in Section 5.2, where minimal utility with the pSEHR dataset was achieved, a hypothesis was made to move this dissertation's work forward. Consid-

ering the research question, this hypothesis made sure we understood if the UMLS-based weak annotations were pushing down NER performance (lower SEHR utility). The hypothesis states the following:

**Hypothesis:** "Assuming the SEHR utility, in the context of the SDG pipeline, is affected by the SEHR document generation and annotation pipeline, then considering the initial approach, it can be said that utility is being pushed down more by the annotation pipeline rather than the SEHR document generation."

With such a hypothesis, more questions than answers arose. The main question was whether an alternative to the UMLS-based weak annotation tool could compete with expert-level annotations like those in the SemClinBr benchmark. The chosen alternative was to use an LLM with SOTA performance, specifically GPT-4o, to annotate SEHRs for the desired STYs.

We isolated the effects of SEHR text and annotations using a comparative study. ChatGPT with the chosen LLM was used to generate annotations for a sample of SEHRs. We annotated 1,200 pSEHRs using a UMLS-based weak annotation tool. So, we randomly chose 35 pSEHRs to be annotated using GPT-4o. We ran this experiment with ChatGPT's annotations to check utility gains, with the assumption that the same trend would hold for larger samples. Thus, those 35 pSEHRs annotated with GPT-4o would be compared with the same ones, however with the original weak annotations from UMLS.

Such a shift to an LLM annotator also required a change in Prompt 5. The most relevant shift with choosing GPT-4o to annotate SEHRs was deciding to replace a FS prompting structure with a functionality available in ChatGPT that enables a user to upload external files to a request together with the written prompt. Such a tool is called File Search.

The file search feature retrieves contextual information from external user-uploaded data to enhance prompts and improve model responses. It operates similarly to a RAG together with ChatGPT (by uploading to the chat interface) and with OpenAI API using the Assistants API. The external knowledge is parsed and chunked, with default settings using 800-token chunks and an overlap of 400 tokens between chunks. It retrieves a maximum of 20 chunks by selecting the most useful ones, and uses their embeddings to enhance the user's input for a more tailored and accurate output.

However, GPT-4o was not trained end-to-end with a retriever, like in the original RAG paper. Also, in the original RAG paper file uploads by users were not possible. Therefore, while it can be considered a RAG tool, it is not a strict RAG model. File Search augments models like GPT-4o with any external files the user provides without requiring end-to-end training with a retriever. Notably, it can retrieve both keyword and semantic context from external files.

We replaced FSL with File Search as a RAG tool. We replaced the 2S examples of Prompt 5 with 4 annotation files from the SemClinBr benchmark. To prevent data leakage, those 4 EHRs

came from the training set of 600 EHRs used for NER. Details on the data split are explored in Section 4.3.4.1. With those changes, it was required to change the written prompt. Prompt 5's text was adjusted and augmented with one pSEHR out of the sample of 35. The final prompt for the comparative study is now "Prompt 35 pSEHRs + GPT-4o". The other sample of pSEHRs is called 35 pSEHRs + UMLS.

We conducted additional work beyond the comparative study to achieve the best results aligned with our research question. We adjusted Prompt 35 pSEHRs + GPT-4o to enable GPT-4o to act as both a SEHR document generator and annotator. GPT-4o generated and annotated 35 SEHRs under this setting to ensure a fair comparison. This is referred to as Prompt 35 SEHRs GPT-4o. We also used a sample of 35 bSEHRs. This sample was called 35 bSEHRs + UMLS. For these samples, ChatGPT was used as well.

#### 4.2.3.2 Further SDG adjustments and final results for SemClinBr

In light of promising results from the comparative study and the performance of 35 SEHRs GPT-4o in terms of utility, we made further improvements to Prompt 35 SEHRs GPT-4o. Such adjustment was driven by the request for a budget to use the OpenAI API to enhance the SDG pipeline capacity for generating and annotating more SEHRs. As a result, we dedicated additional time improving the prompt, including selecting better training EHRs from SemClinBr to use in File Search. From a sample of the 600 EHRs, EHRs with the chosen STYs for NER and with a relatively large amount of annotations were selected. A total of 4 EHRs were used in File Search.

We made a second significant change by adjusting the written prompt. In addition to user requests, we allowed specific instructions to guide the LLM's performance in a conversation or thread. These system instructions gathered details about STY definitions, desired number of annotations per SEHR, and file formats to be returned by the model, providing GPT-4o with valuable guidance not available from the RAG system.

One last change was the language in which both system and user instructions were written. So far, we have been using European Portuguese in the written prompt. However, considering the usage of such SDG pipelines for other applications or by other developers, the user's instructions were adapted to English. The system instructions were written directly in English.

We've been using GPT-4o with the CONCEPT component. We created two new prompts, one with CONCEPT and one without. We called those prompts Prompt GPT-4o adjusted and Prompt GPT-4o adjusted v2, respectively. We found better NER performance with lower privacy risks with Prompt GPT-4o adjusted. Thus, we successfully improved Prompt 35 SEHRs GPT-4o to spend the budget. Next, we detail how the models and hyperparameters were selected, specifically, temperature and top-p threshold.

In August 2024, OpenAI had to offer two GPT-4o snapshot models: gpt-4o-2024-05-13 and gpt-4o-2024-08-06. We tested different hyperparameter values for the temperature and the top-p threshold of both LLMs. We found that setting both hyperparameters to 0.5 resulted in more consistent results than other values. For values near 1.0 and around 0.1-0.3, both models generated SEHRs with an inconsistent number of annotations and text length. We instructed both models to

generate 3 SEHRs for each combination of hyperparameters to minimize money spent while still tuning them. We achieved these results, consistent or inconsistent, for both snapshot models.

So, we used gpt-4o-2024-05-13 and gpt-4o-2024-08-06 with temperature and the top-p threshold of 0.5 to generate 600 SEHRs, which were 300 SEHRs for each model. This setup finished our approach to generate SEHRs to augment SemClinBr. We finished the SDG pipeline with Prompt GPT-4o adjusted and gpt-4o-2024-08-06 with the hyperparameters mentioned above.

#### 4.2.3.3 SDG final pipeline with different PLM encoder applied to PortugueseClinicalNER

After completing the SDG pipeline, we tested it with an unbiased NER benchmark, the PortugueseClinicalNER dataset. We made a crucial shift in Section 4.2.3.1 by using an LLM as an annotator, freeing us to augment any NER dataset. Using the UMLS-based weak annotation tool bounded our decision to choose SemClinBr, the only dataset annotated with UMLS's STYs. Applying the finished SDG pipeline to this new NER dataset became possible with adjustments made to "Prompt GPT-4o adjusted". Specifically, the annotations and their definitions, format, external EHR files for File Search, and desired file output format. All other details remained unchanged.

We selected training files like we did to develop Prompt GPT-4o adjusted. We expanded the number of RAG files from 4 to 5. Regarding the CONCEPT component, we adjusted the sample of conditions. PortugueseClinicalNER comprises clinical notes from the Portuguese Society of Neurology. As a result, we downsized the list of possible conditions or diseases to those related to neurology. Examples of conditions are migraines, epilepsy, meningitis, Huntington's disease, and Wilson's disease. These conditions were not retrieved from UMLS, but rather from extensive search for neurological conditions. We did this because there was no STY related to neurological conditions in specific.

Considering the size of the training corpus, 202 EHRs, and also the fact the budget was only considered for SemClinBr, it was decided to generate only a total of 100 SEHRs. These results can be seen in Section 5.5.1.

#### 4.2.4 Resemblance and privacy disclosure

For evaluating the resemblance between real and synthetic EHRs and assessing privacy risks, we used various techniques. These evaluations were conducted only for the final results regarding SemClinBr and PortugueseClinicalNER. Initially, our focus was on the utility of the SEHRs, so considerations such as semantic similarity and privacy risks were postponed. The prevailing view was that evaluating resemblance and privacy disclosure was more pertinent during the final stages of development.

Regarding privacy disclosure, we measured BLEU scores from bigrams to 5-grams of words (BLEU-2, BLEU-3, BLEU-4, and BLEU-5). We compared BLEU scores from different SEHR samples using various GPT-4o prompts or models. We computed BLEU scores using as references training EHRs. We also calculated for validation and test EHRs, but only show for PortugueseClinicalNER. This was the case because with SemClinBr we wanted to compare against other prompts.

In PortugueseClinicalNER there was no other SEHRs to compare with. We used dumbbell plots to visually represent these comparisons, enabling the illustration of two categorical variables and one quantitative variable. This allowed for clear visualization with a maximum of two SEHR groups. An example of these plots can be seen in Figure 5.4.

As for resemblance, real and synthetic sentences were obtained from EHRs and SEHRs, respectively, with sentence embeddings calculated. We used cosine similarity to compare the embeddings and visualized those embeddings using a scatterplot. While t-SNE is used for dimensionality reduction considering the literature review, we also utilized PCA with 2 components. PCA was used because in my master's degree that was the main dimensionality reduction technique taught. In that sense, we also calculated the KL divergences of the latent variables computed by PCA and t-SNE to measure the entropy between their distributions. This was made in PortugueseClinicalNER, and in SemClinBr to compare Prompt GPT-4o adjusted with Prompt GPT-4o adjusted v2 and gpt-4o-2024-05-13 with gpt-4o-2024-08-06.

It's important to consider the choice of the PLM to compute sentence embeddings. For SemClinBr, we used BioBERTpt(all) (Schneider et al. (2020)) without NER fine-tuning, and for PortugueseClinicalNER, we used MediAIBERTina, the smallest version with 900M parameters, not fine-tuned.

## 4.3 Participant extraction

This section will describe processes, techniques, and tools used to extract participants from the narratives. This includes pre-processing SEHRs and benchmark EHRs, experiment schemes, training pipelines, and evaluations. Several steps of this methodology were changed while the work was being developed.

### 4.3.1 NER dataset and STY selection

Two NER datasets were chosen for our dissertation: SemClinBr and PortugueseClinicalNER. One thing to notice is that throughout this section a distinction is made between the two NER benchmarks used, SemClinBr and PortugueseClinicalNER. This can be seen, below, as a demonstration. When no distinction is made it can be assumed that the details are applied to both benchmarks.

**SemClinBr** SemClinBr was the NER dataset chosen mainly because it was the only one found in Portuguese with annotation tags being retrieved from UMLS. Considering that the work plan contemplated developing a UMLS-based weak annotation tool, it was the most obvious dataset choice. The main downside, however, was that the EHRs were not written in European Portuguese, but rather Brazilian Portuguese. Either way, the SDG pipeline was developed using it.

The selection of STYs for SemClinBr was based on the 13 fine-tuned models developed in Schneider et al. (2020). From those, 5 STYs were chosen based on their relevance. By relevance, we mean those that have a higher importance to be annotated, such as symptoms, diseases,

and results. The final STYs were Sign or Symptom, Disease or Syndrome, Laboratory or Test Result, Quantitative Concept, and Diagnostic Procedure.

**PortugueseClinicalNER** As for the second dataset, in the literature review it was found a NER benchmark with EHRs retrieved from the Portuguese Society of Neurology, PortugueseClinicalNER. This was ideal not only because it was written in European Portuguese, but also because we could test the SDG pipeline developed with an unbiased NER benchmark.

We selected STYs based on their strict F1 performance on the test set, the ones with the lowest performance being selected. Baseline performances were computed for all available STYs, with Results (R), Characterization (CH), Condition (C), Additional Observations (OBS), DateTime (DT), and Therapeutics (THER) being chosen. These results can be seen in Section 5.5.2, more specifically in Table 5.45.

### 4.3.2 Experiment schemes

We conducted several experiments following up on the work developed to generate and annotate SEHRs automatically. This work will use the Train Real/Synthetic Test Real (TRSTR) scheme to measure utility. In other words, we measured the utility by checking how the augmentation of the training EHRs with SEHRs changed the performance of NER models. Thus, we can better target the research question regarding performance gains with data augmentation via SEHRs.

#### 4.3.2.1 TRSTR scheme

**SemClinBr** The TRSTR scheme we developed has several variants to measure utility. Mainly, there are three schemes: one where the training data is composed by 600 EHRs from SemClinBr for training, and the other two with 300 and 100 training EHRs. Adding a small number of SEHRs to the 600 real EHRs led to minor performance gains for certain STYs. Thus, to simulate several data shortage scenarios, we assessed the utility of SEHRs by creating the other two TRSTR scheme variants.

Initial results were obtained by using the TRSTR scheme of 600 EHRs with 600 and 1,200 pSEHRs to determine utility levels for each STY considered. From those results, we were able to better decide on the next steps.

We compared four samples of 35 SEHRs, analyzing two in detail for a comparative study. Two samples used UMLS-based weak annotation (35 pSEHRs + UMLS and 35 bSEHRs + UMLS), while the other two used GPT-4o for generation and/or annotation (35 SEHRs GPT-4o and 35 pSEHRs + GPT-4o). More details can be found in-depth in Section 4.2.3.1.

As for the remaining results, we used similar TRSTR schemes, with the main difference being the SEHRs used. When choosing between Prompt GPT-4o adjusted and Prompt GPT-4o adjusted v2, the only TRSTR scheme variant used was 100 real EHRs for training. When comparing the final SEHR datasets generated and annotated with different GPT-4o snapshot models, we employed all three TRSTR scheme variants. A schematic view of the TRSTR variants and how they impact

the augmented training corpus can be seen in Figure 4.2. The training corpus used for baseline performance is achieved with these TRSTR variants as well. However, no SEHRs are added to it.

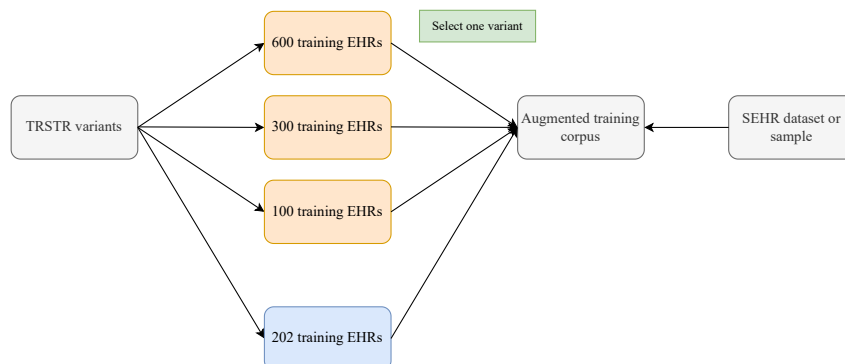


Figure 4.2: Diagram of the TRSTR scheme variants. Regardless of what SEHRs are being used, we selected one of the variants of real EHR samples. Orange refers to the TRSTR variants used with SemClinBr. Blue refers to the TRSTR variant used with PortugueseClinicalNER.

**PortugueseClinicalNER** To test the final SDG pipeline, we included one TRSTR scheme, using 202 EHRs from the PortugueseClinicalNER dataset and 100 SEHRs.

### 4.3.3 Pre-processing

As described in Section 2.3.1, several possible pre-processing and parsing steps can be done so the samples are properly formatted to be given to a model. The pre-processing steps implemented are described below.

#### 4.3.3.1 Sentence segmentation

**SemClinBr** Because the CoNLL annotations created from the SEHRs use the CoNLL-2003 format, each annotation file must have an empty line separating each sentence. We made this step in Section 4.2.2.3 since sentence tokenization is a pre-processing procedure of NER. The spaCy Portuguese sentence boundary detector<sup>4</sup> separated the documents by sentences. Therefore, when reading the CoNLL-2003 text files, the only step was to separate the sentences according to the empty lines in each file.

**PortugueseClinicalNER** After checking the several EHRs available in the benchmark, we noticed that sentences could easily be segmented by checking where the period appeared. Thus, instead of using a sentence tokenizer like the one from spaCy, we separated using the period. We also did this make a change in the pre-processing of the unbiased benchmark.

<sup>4</sup><https://spacy.io/models/pt>

#### 4.3.3.2 Text normalization

**SemClinBr** The only text normalization procedure implemented was case folding, performed simultaneously with tokenization as described in Section 4.3.3.3.

**PortugueseClinicalNER** Case folding was also applied to achieve the results for this NER benchmark.

#### 4.3.3.3 Label alignment & Tokenization

We adjusted the BIO tags twice, before and during tokenization. Our models were tailored to individual STYs, with other STYs being assigned the tag "O". We consider this filtering the STYs, this step being the BIO tag adjustment before tokenization. For example, if the STY chosen for a NER model is "DiseaseOrSyndrome", annotations with labels "SignOrSymptom" or "DiagnosticProcedure" would not be considered entities. Tokenization was performed after separating sentences and filtering the STYs. BioBERTpt and MediAlBERTina utilized the BPE tokenization technique. We used HuggingFace<sup>5</sup>, a valuable NLP tool with numerous publicly available transformer models and tokenizers.

During tokenization, we adjusted the BIO tags to account for the difference between word-level annotations and sub-word level tokens. For example, the participant "cisto renal" had labels "B-DiseaseOrSyndrome, I-DiseaseOrSyndrome" at the word level. Using the BioBERTpt(all) BPE tokenizer, the entity became "ci #sto ren #al," and the labels needed to be adjusted accordingly to "B-DiseaseOrSyndrome, I-DiseaseOrSyndrome, I-DiseaseOrSyndrome, I-DiseaseOrSyndrome". We developed an algorithm to handle three scenarios: when a word is separated into more than one token and either the word's label starts with "B-" or not, and when the token is the word itself.

When a word is split into multiple tokens, the first token will have a label starting with "B-", and subsequent tokens will have labels beginning with "I-". If the word is not an entity or is part of one but not the first word, then all tokens will have the same label as that word. This approach correctly handles "I-" and "O" labels for multi-token words. When a token represents the entire word, its label remains unchanged.

Notice that the special tokens [CLS] and [SEP], as well as their respective labels "O", were added to the beginning and end of the proper arrays. This is a pre-processing step specific for the encoders used to compute the token's contextual embeddings, hence mentioned in Section 4.3.3.4. The complete algorithm can be seen in Algorithm 3.

---

<sup>5</sup><https://huggingface.co>

---

**Algorithm 3** Tokenization and Label Adjustment
 

---

**Input:** labeled sentences, tokenizer  
**Output:** list with (token strings, token IDs, adjusted labels)  
**Initialize:** empty list for the tokenized data  
**for** each sentence in labeled sentences **do**  
   Add to the token strings [CLS] and "O" to the labels  
   Initialize token IDs list  
   **for** each (word, label) in sentence **do**  
     Apply tokenizer to word  
     Add subwords to token strings  
     **if** one word was separated to more than one token **then**  
       **if** label starts with "B-" **then**  
         Extend labels with label followed by I-tag for each subword  
       **else**  
         Extend labels with label for all subwords (either I-tag or O)  
       **end if**  
     **else**  
       Append the label for the single subword  
     **end if**  
   **end for**  
   Add to the token strings [SEP] and "O" to the labels  
   Convert token strings to token IDs  
   Add (token strings, token IDs, labels) to the list for the tokenized data  
**end for**

---

#### 4.3.3.4 Model & training specific pre-processing steps

To ensure consistent sentence length for back-propagation with mini-batch gradient descent, we pad sentences with a special token, [PAD], at the end of a sentence and after the [SEP] special token. These tokens have no impact on the sentence's meaning. Thus, an attention mask is created to set the encoder's attention weights to zero for the [PAD] tokens. This mask excludes [PAD] losses from back-propagating during training.

#### 4.3.4 Training Pipeline

This section will emphasize the details of the training pipeline, from how exactly the data was split and processed as well as the architecture details, hyperparameter handling, and model selection for further testing against other models. An overview of the overall work method followed to achieve the NER results shown in Chapter 5 can be seen in Figure 4.3. The details of this diagram, specifically from the pre-processing step onwards, will be explained in this section and in Section 4.3.5 regarding evaluation metrics. This figure is crucial to understand the how the complete training pipeline works.

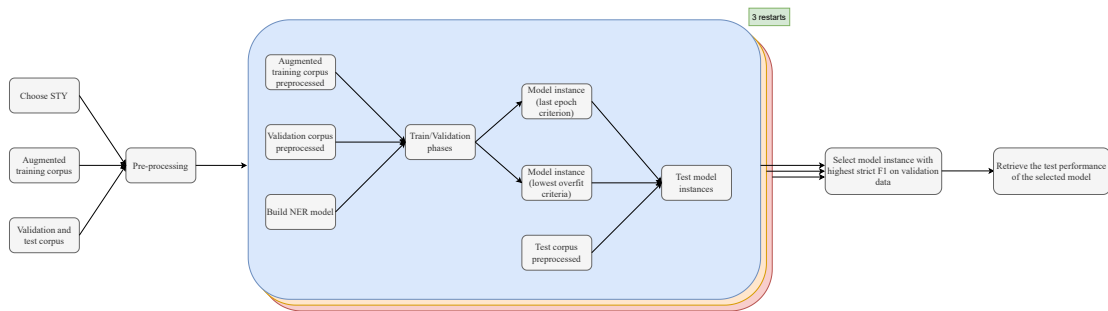


Figure 4.3: Diagram of the overall procedure followed to achieve the test results for each STY and augmented training corpus. The "Augmented training corpus" input is created considering Figure 4.2. For a given STY, the augmented training corpus and validation/test data are preprocessed. For each restart, a training procedure is started and 2 model instances are retrieved. These can either represent a model status for different epochs or for the same one. Test results are retrieved for each of those model instances. After the three restarts, for a maximum of 6 different model instances, the test performance correspondent to the highest validation strict F1 score of all is selected and used in Chapter 5.

To better understand this diagram, in Appendix A.2 is shown three examples with all the results and result selection for a given augmented training corpus and STY.

##### 4.3.4.1 Data splitting

We used a holdout scheme for model validation and testing due to the computational expense of cross-validation with the encoders used, and also considering the literature. To enhance robust-

ness, we shuffled training sequences at each epoch. We chose PyTorch<sup>6</sup> as the DL framework, considering its flexibility and wide range of tools to develop robust training pipelines. We employed PyTorch's DataLoader class to enable sequence shuffling.

**SemClinBr** Our split followed the one in [Schneider et al. \(2020\)](#), where from the SemClinBr dataset with 1,000 documents, the training/validation/testing split was 600/200/200. We used for validation and testing 200 EHRs each. The paper mentioned above had no information regarding which EHRs were used for each split. Therefore, we randomly split the benchmark keeping the seed for reproducibility purposes. Regardless of the TRSTR scheme variants, SEHR datasets or samples, the validation and test EHRs randomly chosen were always the same. The splits for the three TRSTR scheme variants were 600/200/200, 300/200/200, and 100/200/200.

**PortugueseClinicalNER** The usage of a seed to randomly separate the EHRs into training/validation/test sets was the same. What changed, however, was the split itself. Considering the total number of EHRs was 282, the split used was 202/40/40.

#### 4.3.4.2 Hyperparameters

**SemClinBr** We made hyperparameter tuning only for a specific STY, "DiseaseOrSyndrome". We made it that way because it was the first STY we worked on. Also, doing otherwise would increase significantly the computational costs. We only used 600 EHRs in the training corpus for this procedure. We initialized the hyperparameters considering the ones used in [Schneider et al. \(2020\)](#).

Although one could consider that hyperparameter tuning should have been done for each STY, such an approach would only be considered if, for other STYs, overfitting was identified on their respective learning curves. That, however, was not the cause. The best hyperparameter set found was used in all results achieved in our dissertation work for all SemClinBr NER results.

We chose a manual approach over random selection, like grid search, because it involves thoughtful consideration of hyperparameters. For instance, if overfitting is observed, reducing the learning rate, increasing the mini-batch size, increasing the weight decay lambda, and decreasing the number of epochs can help mitigate this issue. The criteria for selecting the best hyperparameters will be described in Section 4.3.4.5.

As for the final hyperparameter set, the optimizer and the learning rate scheduler remained the same as in [Schneider et al. \(2020\)](#). The weight decay regularization has a lambda of 0.1, a mini-batch size 32, a maximum sequence length of 256 (padded), and a learning rate of 3e-5. As for the epochs, a maximum of 4 was considered for fine-tuning.

**PortugueseClinicalNER** As for PortugueseClinicalNER, because no results were published<sup>7</sup> using MediAlbertina, the hyperparameters were initialized with the final set of hyperparameters

---

<sup>6</sup><https://pytorch.org>

<sup>7</sup>Until the writing of this dissertation

used for SemClinBr. We used the same manual approach to reduce overfitting while trying to avoid negatively impacting the performance. This hyperparameter tuning was made mainly for two STYs: Condition and Therapeutics.

As for the hyperparameters, the optimizer and the learning rate scheduler remained the same as in SemClinBr. The weight decay regularization has a lambda of 0.5, a mini-batch size of 8, a maximum sequence length of 128 tokens, and a learning rate  $3e-5$ . As for the epochs, a maximum of 2 was considered for fine-tuning.

#### 4.3.4.3 Model architectures

**SemClinBr** We tried different model architectures, including BioBERTpt(all) with various classification heads, such as BiLSTM with FCNN and CRF layers. Initial tests favored more advanced heads with CRF layers, but for our research on SEHRs, we chose the simplest classification head - a single linear layer - for all experiments. We made this decision because the research question considers performance gains using SEHRs, and not with more advanced classification heads. These tests with different classifier heads were not showcased in our dissertation. On the other hand, adding a CRF layer to our model architecture would be computationally expensive considering the vast amount of models we fine-tuned considering all the restarts, STYs, and TRSTR scheme variants.

**PortugueseClinicalNER** For this benchmark there were no tests regarding different model architectures. The classification head comprises a linear layer. The encoder chosen was MediAlbertina, the European Portuguese version, with 900M parameters. MediAlbertina with 1.5B parameters was not used due to the high computation costs that are required. MediAlbertina was also chosen so that the SDG pipeline would not only be tested on an unbiased NER dataset, but also on an unbiased PLM.

#### 4.3.4.4 Training and validation phases

During each epoch, we employed two phases: training and validation. In the training phase, the loss of a mini-batch is calculated. Back-propagation updates trainable weights and biases with gradients computed by the optimizer. The learning rate scheduler then adjusts the learning rate for the next training step. Each epoch comprises multiple training steps, with the epoch's loss being the average of all training step losses. A training step consists of a parameter update operation. A training procedure with 10 epochs and 1,000 mini-batches has a total of 10,000 training steps, 1,000 per epoch.

The validation phase begins after the final training step, where no further training takes place. The loss is computed without back-propagation. This phase assesses the model's performance on the validation corpus to identify underfitting, overfitting, or both. It's worth noting that a model can exhibit both underfitting and overfitting if its training performance is worse than the baseline's

and its validation performance is considerably worse than its training performance. The model's parameters are stored and a new epoch begins.

After the validation phase on the last epoch, there's a final step before ending the training procedure: selecting a model instance. A model instance consists of the model's weights and biases at the end of each training phase. In other words, we have one model instance per epoch stored at the end of each epoch. Selecting the correct model instance is crucial. If the model performs better in an earlier epoch, it doesn't make sense to use the last model instance. Details on choosing the best model instance from a training procedure are explained in Section 4.3.4.5. At the end, two model instances are returned at the end, one for each model instance selection criteria.

#### 4.3.4.5 Model instance selection

We developed the model instance selection criteria simultaneously with hyperparameter tuning. The model instance selection criteria evolved over time. Initially, no specific criteria were defined, but later we selected the model instance with the lowest validation loss. This approach initially improved results but led to the training loss being much higher than the validation loss.

Therefore, we adjusted the model instance selection criterion. So, instead of selecting the model instance with the lowest validation loss, we changed the criterion to choose the one with the lowest absolute difference between the training and validation losses. The name given to this criteria was lowest overfit. However, this is not the most accurate name because such criteria could select a model instance with a higher training loss than the validation one. Still, this was the name given. The criterion used for hyperparameter tuning was this one.

The lowest validation loss criterion worked well for its purpose, and the lowest overfit criterion worked better. However, in further experiments, the latter criterion consistently produced the second-best model regarding validation evaluation metric scores. Consequently, we added a more straightforward criterion at the end of the validation phase, the last epoch criteria. With the final hyperparameters mentioned in Section 4.3.4.2 for SemClinBr, the model instance for the 4th epoch (last epoch) performed better overall than the lowest overfit criterion. There were times when the opposite occurred. Several times, both criteria returned the same model instance. Thus, each training procedure returned two model instances, one from each model instance selection criteria.

#### 4.3.4.6 Model testing

We applied multiple restarts to all fine-tuned models. Following the approach in the original BERT paper (Devlin et al. (2019)), we randomly restarted the parameters outside of the PLM, namely those in the classification head, to address performance variability for small datasets. The training procedure was restarted three times, selecting the best model instances for each restart based on two criteria detailed in Section 4.3.4.5. The final result for a specific TRSTR variant, sample, and STY was the best model instance from all three training procedures, based on the highest strict F1 on the validation corpus.

**SemClinBr** As mentioned in Section 4.3.4.1, 200 EHRs from the SemClinBr dataset were separated for validation and another 200 EHRs for testing. Emphasizing what is detailed in Section 4.3.2.1, the final test results came from the model with the highest strict F1 on the validation set out of all model restarts and model instance selection criteria.

For example, the final test results for a specific combination of a TRSTR variant, SEHR sample, and STY would be chosen out of 6 model instances, two from each selection criteria for the three restarts.

**PortugueseClinicalNER** As for PortugueseClinicalNER, we followed the same testing procedure as in SemClinBr.

### 4.3.5 Evaluation

Regarding how the models are evaluated, the main classification metrics used were Precision (P), Recall (R), and F1-score (F1). Those metrics will be computed based on the evaluation guidelines defined for the SemEval-2013 Task 9.1 in Segura-Bedmar et al. (2013). Considering our NER models were trained for a specific STY, two of the four sets of scores explained in Section 2.3.4 will have the same metric values, strict and exact scores. So, the scores used in this work were strict evaluation, partial boundary matching, and type matching.

## 4.4 Summary

This chapter focused on explaining the entire work methods, approaches, techniques, procedures, and all their inherent details.

Initially, we used GPT-3.5 to generate 1,200 pSEHRs. However, we suspected the weak annotations from the UMLS tool were limiting utility. To test this, we used GPT-4o to annotate pSEHRs and compared its annotations to those of the UMLS tool. We also used GPT-4o as both generator and annotator of SEHRs. After enhancing the prompt and making adjustments, we applied the final SDG pipeline to the PortugueseClinicalNER dataset. In the participant extraction component, we utilized a TRSTR utility scheme with SemClinBr and PortugueseClinicalNER. Different TRSTR variants were developed to address varying levels of data shortage scenarios. We established a comprehensive training pipeline and utilized metrics based on the scores from the SemEval 2013 task 9.1 for model evaluation. All models were trained with three restarts to address performance instability with small datasets. The best model instance, determined by the highest strict F1 score on the validation data, was chosen to produce the final test results for a given experiment.



# Chapter 5

## Results

This Chapter, as mentioned in Section 1.4, has been targeted with the goal of showcasing the results achieved by following the work methods detailed in Chapter 4.

### 5.1 Outline

This chapter showcases the results from the generation of SEHRs and the extraction of narrative components, specifically participants. The results follow the work methods detailed in Chapter 4.

Section 5.2 shows the initial results obtained, from the initial prompts used with GPT-3.5 to generate the pSEHR dataset to the usage of the UMLS-based weak annotation tool. Its NER results are shown at the end. Section 5.3 focuses on the comparative study made and on the four 35 SEHR samples. Section 5.4 shows the final prompts, resemblance, privacy disclosure, and NER results with GPT-4o using the OpenAI API. SEHR examples considered for these final NER results are shown in Appendix A.1. Section 5.5 consists of testing the final SDG pipeline on an unbiased NER benchmark in European Portuguese, as well as with an unbiased PLM.

### 5.2 Initial approach

This work started with the main approach of generating SEHRs using GPT-3.5 and use a UMLS-based weak annotation to annotate them with the relevant STYs.

This section is divided into Sections 5.2.1, 5.2.2, and 5.2.3, which deal with the SEHR text generation, UMLS-based weak labeling of such SEHRs, and NER, respectively. Section 5.2.1 dives into all the systematic progress made from the most simple prompt to the one that was used. Section 5.2.2 explores the results from the annotation process of the first SEHR dataset generated. Section 5.2.3 shows a better way of defining the baseline performance and shows the first results of an augmented training corpus using the SEHR dataset.

## 5.2.1 SEHRs document generation

This section shows all the developments made towards the generation of SEHRs using GPT-3.5. The prompt engineering process is detailed in Section 5.2.1.1. Evaluation details are shown in Section 5.2.1.2, where an oncology expert assessed a small sample of SEHRs generated for that purpose using ChatGPT.

### 5.2.1.1 Prompt engineering

The results regarding each prompt's template tested are shown in this section. After achieving Prompt 5, the OpenAI API was used with gpt-3.5-turbo.

Regarding the hyperparameter tuning of gpt-3.5-turbo, this was done using BLEU scores. However, it was found a bug in the code that calculated the BLEU scores. We desired SEHRs with lower BLEU scores between them as to have more varied SEHRs. Either way, the hyperparameters used were a temperature of 0.7 and a top-p threshold of 0.9.

**Prompt 1** In European Portuguese, all diseases and conditions, as well as their definitions, were extracted from UMLS. When the CONCEPT does not have a definition, Prompt 1.1 would be used; otherwise, Prompt 1.2 would be the option.

ChatGPT was able to generate SEHRs that were well structured, with several medical terms in European Portuguese, a large number of words, and some level of detail, like symptoms of the CONCEPT.

**[User instructions]** ""Escreve um relatório clínico completo de um cliente qualquer que apresenta leucemia mielode aguda. Nesse relatório clínico devem ser incluídas informações como diagnóstico, plano de tratamentos, sintomas, histórico médico, entre outras informações comuns e cruciais num relatório clínico.""

**Prompt 1.1:** GPT-3.5 prompt with a user instruction given to it and with no condition definition retrieved from UMLS.

**[User instructions]** ""Escreve um relatório clínico completo de um cliente qualquer que apresenta Infecções por Arbovirus, sendo a sua definição a seguinte: "Infecções causadas por vírus transmitidos por artrópodes, gerais ou não especificadas.". Nesse relatório clínico devem ser incluídas informações como diagnóstico, plano de tratamentos, sintomas, histórico médico, entre outras informações comuns e cruciais num relatório clínico.""

**Prompt 1.2:** GPT-3.5 prompt with a user instruction given to it and with a condition definition retrieved from UMLS.

However, Prompts 1.1 and 1.2 had several problems, since real EHRs had a different structure. Although both have demographic data about the patient, the SEHRs had many more segregated

sections, while the real EHRs are less organized and have fewer sections. On the other hand, real EHRs have dates associated with different events, such as exams or next appointments, and quantitative data regarding several aspects of the patient's history, such as heart rate and blood pressure. Real EHRs have misspellings and a much more specific and tailored writing style. This was not the case for the SEHRs generated with Prompts 1.1 and 1.2.

**Prompt 2** Prompt 2 utilized ICL. One real EHR was initially added to the prompt's template (already de-identified). Given this new approach, the prompt's template was adjusted to incorporate FSL properly. Because the prompt became large, the option to concatenate the definition of the CONCEPT, like in Prompt 1.2, was disregarded. The prompt used in this test is shown as Prompt 2.

With this prompt, ChatGPT overcame some issues Prompts 1.1 and 1.2 had, such as the SEHR's structure, misspellings, quantitative data, abbreviations, and writing style. However, the generated SEHRs replicated much of the EHR information in the 1S's example. Even with the CONCEPT component, the output would still mention the disease or condition in the 1S example. Still, when it did not, the rest of the text would be almost a copy of the 1S example, with some quantitative values changed.

**[User instructions]** ""(Exemplo 1): Cria um novo relatório clínico, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com IC descompensada por ITU e DRC Agudizada. Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Resultado 1): ... text of a real EHR with the disease or condition "IC descompensada por ITU e DRC Agudizada"...

(Exemplo 2): Cria um novo relatório clínico, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com Leucemia Mieloide Aguda. Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Resultado 2): ""

**Prompt 2:** GPT-3.5 prompt with a user instruction given to it and with no CONCEPT definition retrieved from UMLS. FS was used, more specifically in a 1S format.

**Prompt 3** For the third prompt, FS with more examples was implemented instead of 1S. Either 2 or 4 EHRs were included as FS examples. The template is the same as Prompt 2 but with more examples.

This time, the results were much better. SEHRs generated with Prompt 3 had a more tailored structure, with dates, quantitative data, exams, and symptoms, as well as misspellings, abbreviations, and the same medical writing style as the real EHRs, which seemed to be adjusted to the CONCEPT given.

Another thing to consider is that, even for the same prompt, ChatGPT generated different SEHRs, sometimes with more abbreviations, others with less quantitative data, different dates and patients, different wording, and the existence or absence of misspellings.

**Prompt 4** In this set of experiments, ICL with prompt engineering was used just like on the two previous prompts, however now we brought back the CONCEPT definition.

Prompt 4 was made to further improve Prompt 3. The idea was to bring back the usage of the definition of a given disease or condition, like in Prompt 1.2, but this time, in a better-structured way. Instead of having only two prompt sections, one for the description of the task and the other for the FS example, an extra section would be added between those two. The prompt is shown below.

**[User instructions]** ""(Exemplo 1): Cria um novo relatório clínico, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com IC descompensada por ITU e DRC Agudizada. Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Conhecimento 1): IC descompensada por ITU e DRC Agudizada - Afecção heterogênea em que o coração é incapaz de bombear sangue suficiente para satisfazer as necessidades metabólicas do corpo. A insuficiência cardíaca pode ser causada por defeitos estruturais, anomalias funcionais (DISFUNÇÃO VENTRICULAR), ou uma sobrecarga súbita além de sua capacidade. A insuficiência cardíaca crônica é mais comum que a insuficiência cardíaca aguda que resulta de injúria repentina à função cardíaca, como INFARTO DO MIOCÁRDIO.

(Resultado 1): ... text of a real EHR with the disease or condition "IC descompensada por ITU e DRC Agudizada"...

(Exemplo 2): Cria um novo relatório clínico, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com pneumonia (PNA). Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Conhecimento 2): pneumonia (PNA) - Infecção do pulmão frequentemente acompanhada por inflamação.

(Resultado 2): ... text of a real EHR with the disease or condition "pneumonia (PNA)"...

(Exemplo 3): Cria um novo relatório clínico, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com Leucemia Mieloide Aguda. Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Conhecimento 3): Leucemia Mieloide Aguda - Expansão clonal de blastos mieloides na medula óssea, sangue e outros tecidos. A leucemia mieloide se desenvolve a partir de mu-

danças nas células, que normalmente produzem NEUTRÓFILOS, BASÓFILOS, EOSINÓFILOS e MONÓCITOS.

(Resultado 3): ""

**Prompt 4:** GPT-3.5 prompt with a user instruction given to it with a condition definition retrieved from UMLS. FS was used, more specifically in a 2S format.

**Prompt 5** This last prompt template was developed after the qualitative evaluation described in Section 5.2.1.2.

Given the healthcare expert's evaluation, the prompt was reduced to a 2S format, and the CONCEPT definitions were removed. This evaluation was crucial because it made the prompt much smaller, considering the small context window gpt-3.5-turbo had.

Two other details were also adjusted. First, the LLM was explicitly told that the text must be written in European Portuguese, and certain abbreviations were replaced by the full text, more specifically, the patient's current status. The prompt's template is the same as Prompt 3, which was adjusted to a 2S format with the aforementioned changes. These changes were made considering the expert's feedback on the SEHRs generated with Prompts 3 and 4.

**[User instructions]** ""(Exemplo 1) Cria um novo relatório clínico em português europeu, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com pneumonia (PNA). Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Resultado 1): ... text of a real EHR with the disease or condition "pneumonia (PNA)"...

(Exemplo 2) Cria um novo relatório clínico em português europeu, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com IC descompensada por ITU e DRC Agudizada. Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Resultado 2): ... text of a real EHR with the disease or condition "IC descompensada por ITU e DRC Agudizada"...

(Exemplo 3) Cria um novo relatório clínico em português europeu, com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com "CONCEPT". Sempre que possível mostrar resultados numéricos que vão de acordo com a doença.

(Resultado 3): ""

**Prompt 5:** GPT-3.5 prompt with a user instruction given to it and with no condition definition retrieved from UMLS. FS was used, more specifically in a 2S format. This was the final prompt achieved for the results in this section.

### 5.2.1.2 Expert’s evaluation

One of the several professionals that collaborated in our work is a healthcare professional with expertise in oncology. This expert reviewed several SEHRs generated using Prompts 3 and 4. Several pairwise comparisons were made, either by changing the prompt’s template regarding the number of FS examples or by using the CONCEPT’s definition or not. The expert did not know from which prompts the several SEHRs came. Given this professional’s expertise, the CONCEPT chosen to generate all SEHRs for this qualitative assessment was Acute Myeloid Leukemia.

In Prompt 3, between 2S or 4S, the difference in the outputs was indistinguishable, according to the expert’s opinion. With Prompt 4, ChatGPT seems to give even more EHR variations than the ones in Prompt 3. The text structure itself seemed to be the same.

Such quality assessment showed that ChatGPT was able to generate good enough EHRs. Besides the appropriate document structure, from a medical point of view, these SEHRs showed proper medication, such as antibiotics, and medical exams. That being said, there were some issues. For example, some reports included the patient status DNR, which stands for Do Not Resuscitate. However, it included future procedures such as surgery, which is contradicting. The other main issue detected was that some terms were in Brazilian Portuguese. This analysis was taken into consideration, and thus we developed Prompt 5.

## 5.2.2 UMLS-based weak annotations

This section focuses on the annotation process results, specifically those of the UMLS-based weak annotation tool, shown in Section 5.2.2.1. This annotation tool was used to annotate the pSEHR dataset annotated using Prompt 5, with a total of 1,200 pSEHRs.

### 5.2.2.1 Raw annotations retrieved from UMLS-based weak annotation tool

We begin by analysing the results of the STYs annotated using the UMLS-based weak annotation tool. This was relevant to check how such tool behaved regarding its annotation capacity across all STYs. Those results can be seen in Table 5.1.

| Semantic Type             | gpt-3.5-turbo |
|---------------------------|---------------|
| Disease or Syndrome       | 4,426         |
| Laboratory or Test Result | 94            |
| Quantitative Concept      | 221           |
| Sign or Symptom           | 5,424         |
| Diagnostic Procedure      | 1,845         |

Table 5.1: Number of occurrences of each STY across the pSEHR dataset consisting of 1,200 SEHRs generated using gpt-3.5-turbo. Up to 5-grams of possible matches were considered.

These results show a clear discrepancy in the UMLS-based weak annotation tool’s ability to annotate certain STYs relative to others. For instance, the developed tool was able to annotate Sign

or Symptom, Disease or Syndrome very well in terms of absolute frequencies. However, Quantitative Concept and Laboratory or Test Result were the STYs with the lowest absolute frequency of entities annotated of all STYs. This could either be due to the SEHR’s text or to the UMLS-based weak annotation tool’s inability to find matches for such STYs in UMLS.

As for the number of sentences retrieved from the pSEHR dataset, the sentence tokenizer computed a total of 21,084 sentences.

### 5.2.3 Participants extraction

We initially considered the possibility of using the NER models fine-tuned in [Schneider et al. \(2020\)](#) for each STY chosen as possible baselines.

However, given their poor performance on our test data split, we decided to train our own baseline models. The poor performance could be due to any pre-processing procedure. We did not look too much into this considering we found a better way to compare NER performance gains. We did so by training our baseline models (TRTR scheme).

#### 5.2.3.1 Baseline models

After measuring the performance of some of the baseline models fine-tuned in [Schneider et al. \(2020\)](#), the results showed a poor performance possible due to different pre-processing procedures. The results are shown in Table 5.2:

|                                                               |    | Strict | Partial | Type   | Loss   |
|---------------------------------------------------------------|----|--------|---------|--------|--------|
| SignOrSymptom ( <a href="#">Schneider et al. (2020)</a> )     | P  | 0.0382 | 0.1893  | 0.3403 |        |
|                                                               | R  | 0.0900 | 0.4455  | 0.8010 | 0.1297 |
|                                                               | F1 | 0.0537 | 0.2657  | 0.4776 |        |
| DiseaseOrSyndrome ( <a href="#">Schneider et al. (2020)</a> ) | P  | 0.0543 | 0.1929  | 0.3316 |        |
|                                                               | R  | 0.1154 | 0.4103  | 0.7051 | 0.0929 |
|                                                               | F1 | 0.0738 | 0.2624  | 0.4511 |        |
| LabOrTestResults ( <a href="#">Schneider et al. (2020)</a> )  | P  | 0.0785 | 0.2278  | 0.3770 |        |
|                                                               | R  | 0.1614 | 0.4684  | 0.7754 | 0.1608 |
|                                                               | F1 | 0.1056 | 0.3065  | 0.5073 |        |
| QuantConcept ( <a href="#">Schneider et al. (2020)</a> )      | P  | 0.2898 | 0.3957  | 0.5015 |        |
|                                                               | R  | 0.4967 | 0.6782  | 0.8596 | 0.0838 |
|                                                               | F1 | 0.3661 | 0.4998  | 0.6334 |        |
| DiagnosticProc ( <a href="#">Schneider et al. (2020)</a> )    | P  | 0.0247 | 0.1621  | 0.2994 |        |
|                                                               | R  | 0.0633 | 0.4152  | 0.7671 | 0.0842 |
|                                                               | F1 | 0.0355 | 0.2331  | 0.4307 |        |

Table 5.2: Baseline models, from [Schneider et al. \(2020\)](#), performance with proper NER classification metrics

Therefore, to ensure the proper baselines were considered when measuring the SEHR’s utility, such models were trained using the pre-processing and splitting procedures developed for this work. 600 EHRs were used to fine-tune BioBERTpt(all). The performance is presented in Table 5.3:

|                              |    | Strict | Partial | Type   | Loss   |
|------------------------------|----|--------|---------|--------|--------|
| SignOrSymptom (baseline)     | P  | 0.4937 | 0.5638  | 0.6339 | 0.0075 |
|                              | R  | 0.6620 | 0.7560  | 0.8500 |        |
|                              | F1 | 0.5656 | 0.6459  | 0.7262 |        |
| DiseaseOrSyndrome (baseline) | P  | 0.5573 | 0.6203  | 0.6834 | 0.0041 |
|                              | R  | 0.7125 | 0.7930  | 0.8736 |        |
|                              | F1 | 0.6254 | 0.6961  | 0.7669 |        |
| LabOrTestResults (baseline)  | P  | 0.3826 | 0.5085  | 0.6344 | 0.0074 |
|                              | R  | 0.4476 | 0.5948  | 0.7421 |        |
|                              | F1 | 0.4126 | 0.5483  | 0.6840 |        |
| QuantConcept (baseline)      | P  | 0.5959 | 0.6582  | 0.7205 | 0.0032 |
|                              | R  | 0.7152 | 0.7900  | 0.8648 |        |
|                              | F1 | 0.6501 | 0.7181  | 0.7861 |        |
| DiagnosticProc (baseline)    | P  | 0.6040 | 0.6475  | 0.6911 | 0.0020 |
|                              | R  | 0.7722 | 0.8278  | 0.8835 |        |
|                              | F1 | 0.6778 | 0.7267  | 0.7756 |        |

Table 5.3: Baseline models, using the pre-processing and data splitting procedures developed in this work, performance with proper NER classification metrics

These results were obtained after proper hyperparameters tuning. To do so, BioBERTpt(all) was fine-tuned for the Disease or Syndrome STY. The lowest overfit criterion was used to find the best set of hyperparameters. The set of hyperparameters used to obtain all TRSTR scheme results is mentioned in Section 4.3.4.2.

It was also considered the possibility of doing hyperparameter tuning for each STY, resulting in a unique mapping between the STY and a set of hyperparameters to be used. However, considering the learning curves obtained for all other STYs, this was unnecessary. For example, in Figures 5.1 and 5.2 are the learning curves obtained from two fine-tuned models in Table 5.3, Diagnostic Procedure, and Quantitative Concept.

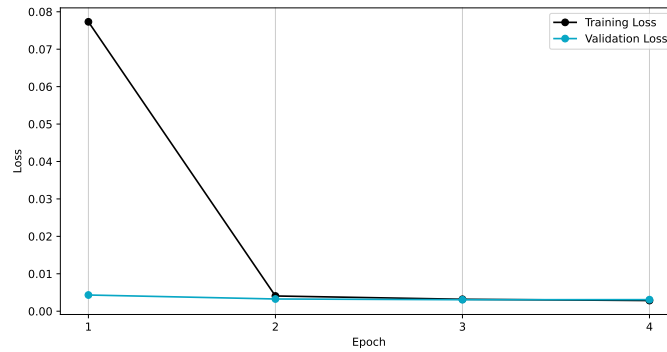


Figure 5.1: Learning curve of BioBERTpt(all) fine-tuned for the Quantitative Concept STY with only 600 EHRs (baseline).

Looking at the learning curves, it can be said that even for different STYs to which the hyperparameters were selected, the training and the validation losses converged. Considering the lowest overfit criterion, the model instance selected in Figures 5.1 and 5.2 would be the one on epoch 3.

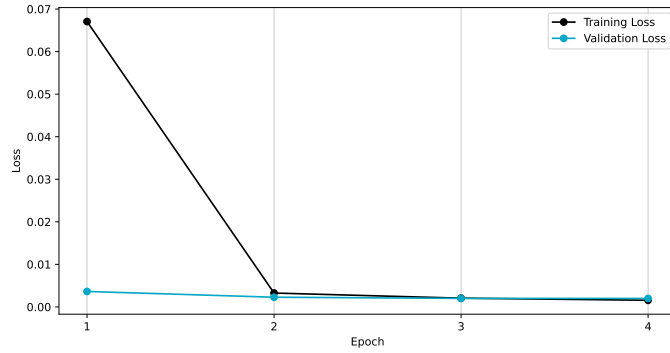


Figure 5.2: Learning curve of BioBERTpt(all) fine-tuned for the Diagnostic Procedure STY with only 600 EHRs (baseline).

Comparing the results from Tables 5.2 and 5.3, the performance gap is noticeable. Despite that, considering that the data-splitting details are unknown, the appropriate baseline choice for this work would always be the ones from the latter table.

### 5.2.3.2 Initial results obtained

Bold results indicate a superior performance was achieved when compared to the baselines. These results consisted of using the TRSTR scheme variant with 600 EHRs augmented with either 600 or 1,200 pSEHRs, part of the pSEHR dataset generated and annotated, according to the previous results shown. The results using 600 pSEHRs are shown in Tables 5.4:

|                                   |    | Strict        | Partial       | Type          | Loss   |
|-----------------------------------|----|---------------|---------------|---------------|--------|
| SignOrSymptom (pSEHRs + UMLS)     | P  | <b>0.5353</b> | <b>0.6036</b> | <b>0.6718</b> |        |
|                                   | R  | 0.6590        | 0.7430        | 0.8270        | 0.0081 |
|                                   | F1 | <b>0.5908</b> | <b>0.6661</b> | <b>0.7414</b> |        |
| DiseaseOrSyndrome (pSEHRs + UMLS) | P  | <b>0.5597</b> | <b>0.6243</b> | <b>0.6889</b> |        |
|                                   | R  | <b>0.7216</b> | <b>0.8049</b> | <b>0.8883</b> | 0.0044 |
|                                   | F1 | <b>0.6304</b> | <b>0.7032</b> | <b>0.7760</b> |        |
| LabOrTestResults (pSEHRs + UMLS)  | P  | 0.3752        | 0.5066        | <b>0.6380</b> |        |
|                                   | R  | 0.4276        | 0.5774        | 0.7271        | 0.0086 |
|                                   | F1 | 0.3997        | 0.5397        | 0.6796        |        |
| QuantConcept (pSEHRs + UMLS)      | P  | <b>0.6007</b> | <b>0.6661</b> | <b>0.7316</b> |        |
|                                   | R  | 0.7100        | 0.7874        | <b>0.8648</b> | 0.0031 |
|                                   | F1 | <b>0.6508</b> | <b>0.7217</b> | <b>0.7926</b> |        |
| DiagnosticProc (pSEHRs + UMLS)    | P  | <b>0.6091</b> | <b>0.6492</b> | <b>0.6893</b> |        |
|                                   | R  | 0.7494        | 0.7987        | 0.8481        | 0.0021 |
|                                   | F1 | 0.6720        | 0.7162        | 0.7605        |        |

Table 5.4: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 600 pSEHRs annotated using the UMLS-based weak annotation tool.

Table 5.4 shows improvements across all scores. However, certain fine-tuned models improved performance with additional 600 pSEHRs. The performance did not change relative to the baselines, except for the Sign or Symptom STY. In this case, we saw a higher relative performance

gain when compared with the other STYs. On the Disease or Syndrome STY, the 600 pSEHRs increased the performance across all metrics.

However, the results are very close to the baseline's. So, the next line of thought was to increase the augmented training corpus with more 600 pSEHRs, for a total of 1,200 pSEHRs. The results are shown in Table 5.5:

|                                   |    | Strict                     | Partial                    | Type                       | Loss   |
|-----------------------------------|----|----------------------------|----------------------------|----------------------------|--------|
| SignOrSymptom (pSEHRs + UMLS)     | P  | <b>0.5404</b> <sup>†</sup> | <b>0.6103</b> <sup>†</sup> | <b>0.6803</b> <sup>†</sup> |        |
|                                   | R  | 0.6490                     | 0.7330                     | 0.8170                     | 0.0084 |
|                                   | F1 | <b>0.5897</b>              | <b>0.6661</b> <sup>†</sup> | <b>0.7424</b> <sup>†</sup> |        |
| DiseaseOrSyndrome (pSEHRs + UMLS) | P  | 0.5532                     | 0.6170                     | 0.6809                     |        |
|                                   | R  | <b>0.7143</b>              | <b>0.7967</b>              | <b>0.8791</b>              | 0.0047 |
|                                   | F1 | 0.6235                     | 0.6954                     | <b>0.7674</b>              |        |
| LabOrTestResults (pSEHRs + UMLS)  | P  | <b>0.3852</b> <sup>†</sup> | 0.5073 <sup>†</sup>        | 0.6294                     |        |
|                                   | R  | 0.4409 <sup>†</sup>        | 0.5807 <sup>†</sup>        | 0.7205                     | 0.0088 |
|                                   | F1 | 0.4112 <sup>†</sup>        | 0.5415 <sup>†</sup>        | 0.6718                     |        |
| QuantConcept (pSEHRs + UMLS)      | P  | <b>0.6105</b> <sup>†</sup> | <b>0.6724</b> <sup>†</sup> | <b>0.7344</b> <sup>†</sup> |        |
|                                   | R  | 0.7113 <sup>†</sup>        | 0.7835                     | 0.8557                     | 0.0032 |
|                                   | F1 | <b>0.6571</b> <sup>†</sup> | <b>0.7237</b> <sup>†</sup> | <b>0.7904</b>              |        |
| DiagnosticProc (pSEHRs + UMLS)    | P  | <b>0.6191</b> <sup>†</sup> | <b>0.6606</b> <sup>†</sup> | <b>0.7021</b> <sup>†</sup> |        |
|                                   | R  | 0.7367                     | 0.7861                     | 0.8354                     | 0.0020 |
|                                   | F1 | 0.6728 <sup>†</sup>        | 0.7179 <sup>†</sup>        | 0.7630 <sup>†</sup>        |        |

<sup>†</sup> Indicates higher performance over BioBERTpt(all) fine-tuned results in Table 5.4.

Table 5.5: BioBERTpt(all) results fine-tuned with an augmented training corpus with 1,200 pSEHRs annotated using the UMLS-based weak annotation tool

These results show several tendencies. Besides the Disease or Syndrome and Laboratory or Test Result STYs, the same metrics as in Table 5.4 achieve higher performance than the baselines. Also, for those metrics, the additional 600 pSEHRs improved the performance further, except the strict F1 for Sign and Symptom and type F1 for Quantitative Concept. However, such an increment of 600 pSEHRs decreased the performance of the Disease and Syndrome STY for all metrics and scores. These results show that the pSEHRs annotated with UMLS do not have the initial hoped-for utility.

### 5.3 Comparative Study and Proof of Concept with an LLM as SEHR generator and annotator

After achieving the first results of this dissertation's work, it became evident that being able to achieve a good utility for SemClinBr across the vast majority of the STYs chosen would be more complicated than initially considered. Indeed, it was clear that developing a SDG pipeline capable of having SEHRs with complex texts and near expert-level annotations would be difficult.

Section 5.3.1 dives into the results achieved regarding the annotation and generation of SEHRs. Section 5.3.2 will use the SEHR sample datasets developed in Section 5.3.1 to answer the hypothesis made while also checking which SEHR sample dataset achieved the highest utility, overall.

### 5.3.1 SEHRs document and annotations generation

Prompt 5 was adjusted to tackle the comparative study and to generate and annotate SEHRs using GPT-4o. Further details can be read in Section 4.2.3.1. The prompt used to generate annotations for the sample of 35 pSEHRs with GPT-4o annotations can be seen below.

**[User instructions]** ""Cria um ficheiro XML de um relatório clínico com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis, relatório esse que vou enviar abaixo, sendo que deves acrescentar as devidas anotações. As tags possíveis são as seguintes: "Diagnostic Procedure", "Disease or Syndrome", "Finding", "Health Care Activity", "Sign or Symptom", "Laboratory or Test Result", "Medical Device", "Pharmacologic Substance", "Quantitative Concept", e "Therapeutic or Preventive Procedure". Sempre que possível mostrar resultados numéricos que vão de acordo com a doença. Como referência envio quatro relatórios clínicos em formato XML.

Relatório clínico: (pSEHR text to be annotated...) ""

**Prompt 35 pSEHRs + GPT-4o:** Prompt for ChatGPT using GPT-4o to annotate a given pSEHR with a user instruction given to it with the CONCEPT retrieved from UMLS. Instead of FS prompting, 4 examples, from the 600 EHR training corpus of SemClinBr, were uploaded to File Search.

Notice that in Prompt 35 pSEHRs + GPT-4o there are more STYs than those chosen to measure utility. This is because at the time it was under consideration the possibility of adding more STYs to our dissertation's work.

As detailed in Section 4.2.3, two more 35 SEHR samples were generated and annotated besides the comparative study ones. The first, Prompt 35 SEHRs GPT-4o, can be seen below.

**[User instructions]** ""Cria um ficheiro XML de um relatório clínico com dados, sintomas, exames, e problemas fictícios mas medicamente razoáveis para um paciente com Doença de Kaschin-Beck envolvendo a perna. As tags possíveis são as seguintes: "Diagnostic Procedure", "Disease or Syndrome", "Finding", "Health Care Activity", "Sign or Symptom", "Laboratory or Test Result", "Medical Device", "Pharmacologic Substance", "Quantitative Concept", e "Therapeutic or Preventive Procedure". Sempre que possível mostrar resultados numéricos que vão de acordo com a doença. Como referência envio quatro relatórios clínicos em formato XML.""

**Prompt 35 SEHRs GPT-4o:** Prompt for ChatGPT using GPT-4o to generate and annotate a SEHR with a user instruction given to it with the CONCEPT retrieved from UMLS. Instead of FS prompting, 4 examples, from the 600 EHR training corpus of SemClinBr, were uploaded to File Search.

The second prompt followed the format of Prompt 5, however, instead of using as FS examples EHRs from Hospital P, those FS examples would come from the training corpus of SemClinBr

directly, just like in Prompt 35 pSEHRs + GPT-4o and Prompt 35 SEHRs GPT-4o. These were the bSEHRs generated. Further details can be read in Section 4.2.1.

As an initial understanding of how the number of STYs were distributed across these four new SEHR samples, occurrences per STY and samples can be seen in Table 5.6.

| Semantic Type             | 35 pSEHRs + UMLS | 35 pSEHRs + GPT-4o | 35 bSEHRs + UMLS | 35 SEHRs GPT-4o |
|---------------------------|------------------|--------------------|------------------|-----------------|
| Disease or Syndrome       | 135              | 88                 | 69               | 44              |
| Laboratory or Test Result | 2                | 164                | 0                | 91              |
| Quantitative Concept      | 6                | 82                 | 9                | 160             |
| Sign or Symptom           | 159              | 165                | 92               | 156             |
| Diagnostic Procedure      | 43               | 42                 | 39               | 66              |

Table 5.6: Number of occurrences of each STY across the four SEHRs samples, either generated by gpt-3.5-turbo or GPT-4o (ChatGPT), as well as either annotated using the UMLS-based weak annotation tool or GPT-4o (ChatGPT).

Looking at these results, mainly at the two 35 SEHRs samples directly involved in the comparative study, it is very noticeable that, for the same 35 pSEHRs generated by gpt-3.5-turbo, ChatGPT with GPT-4o can annotate much more entities relative to the UMLS-based weak annotation tool. Such analysis can be better understood by looking at the occurrences of Laboratory or Test Result and Quantitative Concept between 35 pSEHRs + UMLS and 35 pSEHRs + GPT-4o. As for the other STYs, GPT-4o annotates less Disease or Syndrome than UMLS, while for Sign or Symptom and Diagnostic Procedure the number of annotations is similar.

Regarding 35 bSEHRs + UMLS, it has the lowest number of annotations across all four SEHRs samples for the majority of STYs, except for Quantitative Concept and Disease or Syndrome. In fact, the UMLS-based weak annotation tool was not able to annotate any entity with Laboratory or Test Result as STY. It was also the only SEHRs sample with less than 100 annotations for Sign or Symptom.

As for 35 SEHRs GPT-4o, despite GPT-4o not being able to generate and annotate more entities for Disease or Syndrome and Laboratory or Test Result, it generated much more annotations for Quantitative Concept and Diagnostic Procedure.

### 5.3.2 Participant extraction

The initial results in Section 5.2.3.2 showed a small increase in the strict F1 for all STYs overall. However, the desire was to go further and try to improve the SEHR's utility more. The question was how to go about it. First, it was required to pinpoint what could be affecting the utility of the SEHRs. This section shows the NER results achieved with the four SEHR samples generated and annotated.

Also, to better understand the utility of the SEHRs, all three TRSTR scheme variants were applied. Mainly, a training corpus composed of 600, 300, and 100 real EHRs. More details can be read in Section 4.3.2.1.

### 5.3.2.1 Sign or Symptom

The results using 600, 300, and 100 EHRs with different SEHR samples are shown in Tables 5.7, 5.8, and 5.9, respectively.

|                                    |    | Strict                     | Partial             | Type                | Loss   |
|------------------------------------|----|----------------------------|---------------------|---------------------|--------|
| SignOrSymptom (baseline)           | P  | 0.4937                     | 0.5638              | 0.6339              |        |
|                                    | R  | 0.6620                     | 0.7560              | 0.8500              | 0.0075 |
|                                    | F1 | 0.5656                     | 0.6459              | 0.7262              |        |
| SignOrSymptom (35 pSEHRs + UMLS)   | P  | 0.4970                     | 0.5694              | 0.6418              |        |
|                                    | R  | 0.6520                     | 0.7470              | 0.8420              | 0.0079 |
|                                    | F1 | 0.5640                     | 0.6462              | 0.7284              |        |
| SignOrSymptom (35 bSEHRs + UMLS)   | P  | 0.4948                     | 0.5644              | 0.6341              |        |
|                                    | R  | 0.6680                     | 0.7620              | 0.8560              | 0.0074 |
|                                    | F1 | 0.5685                     | 0.6485              | 0.7285              |        |
| SignOrSymptom (35 SEHRs GPT-4o)    | P  | 0.5217                     | 0.5881              | <b>0.6545</b>       |        |
|                                    | R  | <b>0.6840</b>              | <b>0.7710</b>       | <b>0.8580</b>       | 0.0074 |
|                                    | F1 | 0.5920                     | <b>0.6672</b>       | <b>0.7425</b>       |        |
| SignOrSymptom (35 pSEHRs + GPT-4o) | P  | <b>0.5260</b> <sup>†</sup> | 0.5900 <sup>†</sup> | 0.6540 <sup>†</sup> |        |
|                                    | R  | 0.6780 <sup>†</sup>        | 0.7605 <sup>†</sup> | 0.8430 <sup>†</sup> | 0.0074 |
|                                    | F1 | <b>0.5924</b> <sup>†</sup> | 0.6645 <sup>†</sup> | 0.7366 <sup>†</sup> |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.7: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Overall, using GPT-4o to generate and annotate or only to annotate SEHRs increases the performance of this STY. In terms of best performance across all fine-tuned models, the usage of 300 EHRs showed some disparity. Even when using UMLS to annotate, the bSEHRs achieved the best recall for all scores. Although with 600 EHRs, the 35 SEHRs GPT-4o synthetic corpus achieved the highest utility, when facing a much smaller sample of EHRs, it fell short to the 35 pSEHRs + GPT-4o setup.

|                                    |    | Strict                     | Partial                    | Type                       | Loss   |
|------------------------------------|----|----------------------------|----------------------------|----------------------------|--------|
| SignOrSymptom (baseline)           | P  | 0.3929                     | 0.4900                     | 0.5871                     |        |
|                                    | R  | 0.5280                     | 0.6585                     | 0.7890                     | 0.0091 |
|                                    | F1 | 0.4505                     | 0.5619                     | 0.6732                     |        |
| SignOrSymptom (35 pSEHRs + UMLS)   | P  | 0.3861                     | 0.4842                     | 0.5824                     |        |
|                                    | R  | 0.5270 <sup>†</sup>        | 0.6610 <sup>†</sup>        | 0.7950 <sup>†</sup>        | 0.0088 |
|                                    | F1 | 0.4457                     | 0.5590                     | 0.6723                     |        |
| SignOrSymptom (35 bSEHRs + UMLS)   | P  | 0.3884                     | 0.4837                     | 0.5790                     |        |
|                                    | R  | <b>0.5360</b>              | <b>0.6675</b>              | <b>0.7990</b>              | 0.0087 |
|                                    | F1 | 0.4504                     | 0.5609                     | 0.6714                     |        |
| SignOrSymptom (35 SEHRs GPT-4o)    | P  | 0.4142                     | 0.5094                     | 0.6045                     |        |
|                                    | R  | 0.5310                     | 0.6530                     | 0.7750                     | 0.0089 |
|                                    | F1 | <b>0.4654</b>              | 0.5723                     | 0.6792                     |        |
| SignOrSymptom (35 pSEHRs + GPT-4o) | P  | <b>0.4171</b> <sup>†</sup> | <b>0.5169</b> <sup>†</sup> | <b>0.6167</b> <sup>†</sup> |        |
|                                    | R  | 0.5180                     | 0.6420                     | 0.7660                     | 0.0088 |
|                                    | F1 | 0.4621 <sup>†</sup>        | <b>0.5727</b> <sup>†</sup> | <b>0.6833</b> <sup>†</sup> |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.8: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

When comparing the setups 35 pSEHRs + UMLS and 35 pSEHRs + GPT-4o, it is noticeable that the GPT-4o annotations have a higher utility than the UMLS-based annotations. This is even more visible when facing the TRSTR variant with 100 EHRs.

|                                    |    | Strict                    | Partial                   | Type                      | Loss   |
|------------------------------------|----|---------------------------|---------------------------|---------------------------|--------|
| SignOrSymptom (baseline)           | P  | 0.2180                    | 0.4087                    | 0.5995                    |        |
|                                    | R  | 0.1600                    | 0.3000                    | 0.4400                    | 0.0131 |
|                                    | F1 | 0.1845                    | 0.3460                    | 0.5075                    |        |
| SignOrSymptom (35 pSEHRs + UMLS)   | P  | 0.2861                    | 0.4539                    | <b>0.6217<sup>†</sup></b> |        |
|                                    | R  | 0.2420                    | 0.3840                    | 0.5260                    | 0.0120 |
|                                    | F1 | 0.2622                    | 0.4160                    | 0.5699                    |        |
| SignOrSymptom (35 bSEHRs + UMLS)   | P  | 0.2451                    | 0.4300                    | 0.6148                    |        |
|                                    | R  | 0.1260                    | 0.2210                    | 0.3160                    | 0.0131 |
|                                    | F1 | 0.1664                    | 0.2919                    | 0.4174                    |        |
| SignOrSymptom (35 SEHRs GPT-4o)    | P  | 0.3086                    | 0.4575                    | 0.6063                    |        |
|                                    | R  | 0.2830                    | 0.4195                    | 0.5560                    | 0.0123 |
|                                    | F1 | 0.2953                    | 0.4377                    | 0.5801                    |        |
| SignOrSymptom (35 pSEHRs + GPT-4o) | P  | <b>0.3520<sup>†</sup></b> | <b>0.4868<sup>†</sup></b> | 0.6216                    |        |
|                                    | R  | <b>0.3200<sup>†</sup></b> | <b>0.4425<sup>†</sup></b> | <b>0.5650<sup>†</sup></b> | 0.0114 |
|                                    | F1 | <b>0.3353<sup>†</sup></b> | <b>0.4636<sup>†</sup></b> | <b>0.5919<sup>†</sup></b> |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.9: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

### 5.3.2.2 Disease or Syndrome

The results using 600, 300, and 100 EHRs with different SEHR samples are shown in Tables 5.10, 5.11, and 5.12, respectively.

|                                        |    | Strict              | Partial             | Type                | Loss   |
|----------------------------------------|----|---------------------|---------------------|---------------------|--------|
| DiseaseOrSyndrome (baseline)           | P  | <b>0.5573</b>       | <b>0.6203</b>       | <b>0.6834</b>       |        |
|                                        | R  | <b>0.7125</b>       | 0.7930              | 0.8736              | 0.0041 |
|                                        | F1 | <b>0.6254</b>       | <b>0.6961</b>       | <b>0.7669</b>       |        |
| DiseaseOrSyndrome (35 pSEHRs + UMLS)   | P  | 0.5331              | 0.5999              | 0.6667              |        |
|                                        | R  | 0.7088 <sup>†</sup> | 0.7976 <sup>†</sup> | 0.8864 <sup>†</sup> | 0.0039 |
|                                        | F1 | 0.6085              | 0.6847              | 0.7610 <sup>†</sup> |        |
| DiseaseOrSyndrome (35 bSEHRs + UMLS)   | P  | 0.5358              | 0.6033              | 0.6708              |        |
|                                        | R  | <b>0.7125</b>       | <b>0.8022</b>       | <b>0.8919</b>       | 0.0040 |
|                                        | F1 | 0.6116              | 0.6887              | 0.7657              |        |
| DiseaseOrSyndrome (35 SEHRs GPT-4o)    | P  | 0.5538              | 0.6177              | 0.6817              |        |
|                                        | R  | 0.6978              | 0.7784              | 0.8590              | 0.0040 |
|                                        | F1 | 0.6175              | 0.6888              | 0.7601              |        |
| DiseaseOrSyndrome (35 pSEHRs + GPT-4o) | P  | 0.5416 <sup>†</sup> | 0.6072 <sup>†</sup> | 0.6728 <sup>†</sup> |        |
|                                        | R  | 0.7033              | 0.7885              | 0.8736              | 0.0041 |
|                                        | F1 | 0.6120 <sup>†</sup> | 0.6861 <sup>†</sup> | 0.7602              |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.10: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Observing the results achieved, the baseline results were harder to outperform for this STY. Using 600 EHRs, only the sample with bSEHRs could outperform the baseline, in this case for all recall metrics. Although the remaining results improved over the baseline, the best performance went towards the sample 35 pSEHRs + UMLS. For Disease or Syndrome, the GPT-4o setups could not outperform those samples across most of the metrics and scores.

|                                        |    | Strict                     | Partial                    | Type                       | Loss   |
|----------------------------------------|----|----------------------------|----------------------------|----------------------------|--------|
| DiseaseOrSyndrome (baseline)           | P  | <b>0.4937</b>              | <b>0.5789</b>              | <b>0.6640</b>              | 0.0055 |
|                                        | R  | 0.5733                     | 0.6722                     | 0.7711                     |        |
|                                        | F1 | 0.5305                     | 0.6220                     | 0.7136                     |        |
| DiseaseOrSyndrome (35 pSEHRs + UMLS)   | P  | 0.4764 <sup>†</sup>        | 0.5558 <sup>†</sup>        | 0.6352 <sup>†</sup>        | 0.0047 |
|                                        | R  | <b>0.6099</b> <sup>†</sup> | 0.7115                     | 0.8132                     |        |
|                                        | F1 | <b>0.5349</b> <sup>†</sup> | <b>0.6241</b> <sup>†</sup> | 0.7133 <sup>†</sup>        |        |
| DiseaseOrSyndrome (35 bSEHRs + UMLS)   | P  | 0.4724                     | 0.5604                     | 0.6483                     | 0.0050 |
|                                        | R  | 0.5806                     | 0.6886                     | 0.7967                     |        |
|                                        | F1 | 0.5210                     | 0.6179                     | <b>0.7149</b>              |        |
| DiseaseOrSyndrome (SEHRs GPT-4o)       | P  | 0.4907                     | 0.5743                     | 0.6579                     | 0.0054 |
|                                        | R  | 0.5806                     | 0.6795                     | 0.7784                     |        |
|                                        | F1 | 0.5319                     | 0.6225                     | 0.7131                     |        |
| DiseaseOrSyndrome (35 pSEHRs + GPT-4o) | P  | 0.4616                     | 0.5425                     | 0.6234                     | 0.0041 |
|                                        | R  | 0.6062                     | <b>0.7125</b> <sup>†</sup> | <b>0.8187</b> <sup>†</sup> |        |
|                                        | F1 | 0.5241                     | 0.6160                     | 0.7078                     |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.11: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Regarding the 35 pSEHRs + UMLS and 35 pSEHRs + GPT-4o setups, it can be seen that the former outperforms the latter, in general, for this STY. This was not the case only when using 600 EHRs, shown in Table 5.10.

|                                        |    | Strict                     | Partial                    | Type                       | Loss   |
|----------------------------------------|----|----------------------------|----------------------------|----------------------------|--------|
| DiseaseOrSyndrome (baseline)           | P  | 0.3151                     | 0.5034                     | <b>0.6918</b>              | 0.0081 |
|                                        | R  | 0.0842                     | 0.1346                     | 0.1850                     |        |
|                                        | F1 | 0.1329                     | 0.2124                     | 0.2919                     |        |
| DiseaseOrSyndrome (35 pSEHRs + UMLS)   | P  | 0.3472                     | 0.4706                     | 0.5939                     | 0.0071 |
|                                        | R  | <b>0.3352</b> <sup>†</sup> | <b>0.4542</b> <sup>†</sup> | <b>0.5733</b> <sup>†</sup> |        |
|                                        | F1 | <b>0.3411</b> <sup>†</sup> | <b>0.4623</b> <sup>†</sup> | <b>0.5834</b> <sup>†</sup> |        |
| DiseaseOrSyndrome (35 bSEHRs + UMLS)   | P  | 0.2299                     | 0.3705                     | 0.5112                     | 0.0080 |
|                                        | R  | 0.1886                     | 0.3040                     | 0.4194                     |        |
|                                        | F1 | 0.2072                     | 0.3340                     | 0.4608                     |        |
| DiseaseOrSyndrome (35 SEHRs GPT-4o)    | P  | 0.3059                     | 0.4668                     | 0.6277                     | 0.0073 |
|                                        | R  | 0.2106                     | 0.3214                     | 0.4322                     |        |
|                                        | F1 | 0.2495                     | 0.3807                     | 0.5119                     |        |
| DiseaseOrSyndrome (35 pSEHRs + GPT-4o) | P  | <b>0.3715</b> <sup>†</sup> | <b>0.5154</b> <sup>†</sup> | 0.6592 <sup>†</sup>        | 0.0072 |
|                                        | R  | 0.2436                     | 0.3379                     | 0.4322                     |        |
|                                        | F1 | 0.2942                     | 0.4082                     | 0.5221                     |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.12: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

### 5.3.2.3 Laboratory or Test Result

The results using 600, 300, and 100 EHRs with different SEHR samples are shown in Tables 5.13, 5.14, and 5.15, respectively.

|                                      |    | Strict                    | Partial                   | Type                      | Loss   |
|--------------------------------------|----|---------------------------|---------------------------|---------------------------|--------|
| LabOrTestResult (baseline)           | P  | 0.3826                    | 0.5085                    | 0.6344                    |        |
|                                      | R  | 0.4476                    | 0.5948                    | 0.7421                    | 0.0074 |
|                                      | F1 | 0.4126                    | 0.5483                    | 0.6840                    |        |
| LabOrTestResult (35 pSEHRs + UMLS)   | P  | 0.3716                    | 0.4993                    | 0.6270                    |        |
|                                      | R  | 0.4309                    | 0.5790                    | 0.7271                    | 0.0072 |
|                                      | F1 | 0.3991                    | 0.5362                    | 0.6733                    |        |
| LabOrTestResult (35 bSEHRs + UMLS)   | P  | <b>0.3979</b>             | <b>0.5238</b>             | <b>0.6498</b>             |        |
|                                      | R  | 0.4443                    | 0.5849                    | 0.7255                    | 0.0073 |
|                                      | F1 | 0.4198                    | 0.5527                    | 0.6855                    |        |
| LabOrTestResult (35 SEHRs GPT-4o)    | P  | 0.3677                    | 0.4943                    | 0.6209                    |        |
|                                      | R  | 0.4276                    | 0.5749                    | 0.7221                    | 0.0073 |
|                                      | F1 | 0.3954                    | 0.5315                    | 0.6677                    |        |
| LabOrTestResult (35 pSEHRs + GPT-4o) | P  | 0.3906 <sup>†</sup>       | 0.5107 <sup>†</sup>       | 0.6309 <sup>†</sup>       |        |
|                                      | R  | <b>0.4842<sup>†</sup></b> | <b>0.6331<sup>†</sup></b> | <b>0.7820<sup>†</sup></b> | 0.0068 |
|                                      | F1 | <b>0.4324<sup>†</sup></b> | <b>0.5654<sup>†</sup></b> | <b>0.6984<sup>†</sup></b> |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.13: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Looking at the best results across all TRSTR variants, in general, the best-performing setup is 35 pSEHRs + GPT-4o. This is not the case, however, in Table 5.14, where the baseline model presents the best performance overall. Yet, the fine-tuned model achieved the second-best performance using the aforementioned SEHR sample. One other detail to notice is the difference between the test losses. On all three tables, the 35 pSEHRs + GPT-4o setup always achieves the lowest test loss, with a visible loss gap relative to all other fine-tuned models on Tables 5.13 and 5.15.

|                                      |    | Strict                    | Partial             | Type                | Loss   |
|--------------------------------------|----|---------------------------|---------------------|---------------------|--------|
| LabOrTestResult (baseline)           | P  | 0.3424                    | 0.4694              | 0.5964              |        |
|                                      | R  | <b>0.4376</b>             | <b>0.5998</b>       | <b>0.7621</b>       | 0.0070 |
|                                      | F1 | <b>0.3842</b>             | <b>0.5267</b>       | 0.6691              |        |
| LabOrTestResult (35 pSEHRs + UMLS)   | P  | 0.3298                    | 0.4826              | 0.6354 <sup>†</sup> |        |
|                                      | R  | 0.3627                    | 0.5308              | 0.6988              | 0.0083 |
|                                      | F1 | 0.3455                    | 0.5055              | 0.6656 <sup>†</sup> |        |
| LabOrTestResult (35 bSEHRs + UMLS)   | P  | 0.3411                    | <b>0.4992</b>       | 0.6572              |        |
|                                      | R  | 0.3394                    | 0.4967              | 0.6539              | 0.0089 |
|                                      | F1 | 0.3403                    | 0.4979              | 0.6555              |        |
| LabOrTestResult (35 SEHRs GPT-4o)    | P  | 0.3249                    | 0.4921              | <b>0.6593</b>       |        |
|                                      | R  | 0.3428                    | 0.5191              | 0.6955              | 0.0089 |
|                                      | F1 | 0.3336                    | 0.5053              | <b>0.6769</b>       |        |
| LabOrTestResult (35 pSEHRs + GPT-4o) | P  | <b>0.3482<sup>†</sup></b> | 0.4881 <sup>†</sup> | 0.6280              |        |
|                                      | R  | 0.3894 <sup>†</sup>       | 0.5458 <sup>†</sup> | 0.7022 <sup>†</sup> | 0.0083 |
|                                      | F1 | 0.3676 <sup>†</sup>       | 0.5153 <sup>†</sup> | 0.6630              |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.14: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Tackling the comparative study, the 35 pSEHRs + GPT-4o setup results are visibly higher than the 35 pSEHRs + UMLS performance. The GPT-4o annotations achieved a higher utility for all metrics except for type P and type F1 in Table 5.14.

|                                      |    | Strict                     | Partial                    | Type                       | Loss   |
|--------------------------------------|----|----------------------------|----------------------------|----------------------------|--------|
| LabOrTestResult (baseline)           | P  | 0.2220                     | 0.4260                     | 0.6300                     |        |
|                                      | R  | 0.1747                     | 0.3353                     | 0.4958                     | 0.0093 |
|                                      | F1 | 0.1955                     | 0.3752                     | 0.5549                     |        |
| LabOrTestResult (35 pSEHRs + UMLS)   | P  | 0.2393                     | 0.4065                     | 0.5738                     |        |
|                                      | R  | 0.2130                     | 0.3619                     | 0.5108                     | 0.0101 |
|                                      | F1 | 0.2254                     | 0.3829                     | 0.5405                     |        |
| LabOrTestResult (35 bSEHRs + UMLS)   | P  | 0.2106                     | 0.3991                     | 0.5876                     |        |
|                                      | R  | 0.1581                     | 0.2995                     | 0.4409                     | 0.0104 |
|                                      | F1 | 0.1806                     | 0.3422                     | 0.5038                     |        |
| LabOrTestResult (35 SEHRs GPT-4o)    | P  | 0.2377                     | <b>0.4420</b>              | <b>0.6464</b>              |        |
|                                      | R  | 0.1364                     | 0.2537                     | 0.3710                     | 0.0103 |
|                                      | F1 | 0.1734                     | 0.3224                     | 0.4715                     |        |
| LabOrTestResult (35 pSEHRs + GPT-4o) | P  | <b>0.2547</b> <sup>†</sup> | 0.4270 <sup>†</sup>        | 0.5993 <sup>†</sup>        |        |
|                                      | R  | <b>0.2496</b> <sup>†</sup> | <b>0.4185</b> <sup>†</sup> | <b>0.5874</b> <sup>†</sup> | 0.0089 |
|                                      | F1 | <b>0.2521</b> <sup>†</sup> | <b>0.4227</b> <sup>†</sup> | <b>0.5933</b> <sup>†</sup> |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.15: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

#### 5.3.2.4 Quantitative Concept

The results using 600, 300, and 100 EHRs with different SEHR samples are shown in Tables 5.16, 5.17, and 5.18, respectively.

|                                   |    | Strict              | Partial             | Type                       | Loss   |
|-----------------------------------|----|---------------------|---------------------|----------------------------|--------|
| QuantConcept (baseline)           | P  | 0.5959              | 0.6582              | 0.7205                     |        |
|                                   | R  | 0.7152              | 0.7900              | 0.8648                     | 0.0032 |
|                                   | F1 | 0.6501              | 0.7181              | 0.7861                     |        |
| QuantConcept (35 pSEHRs + UMLS)   | P  | 0.5905              | 0.6595              | <b>0.7284</b> <sup>†</sup> |        |
|                                   | R  | 0.7126 <sup>†</sup> | 0.7958 <sup>†</sup> | <b>0.8791</b> <sup>†</sup> | 0.0031 |
|                                   | F1 | 0.6458              | 0.7213 <sup>†</sup> | <b>0.7967</b> <sup>†</sup> |        |
| QuantConcept (35 bSEHRs + UMLS)   | P  | 0.5890              | 0.6521              | 0.7152                     |        |
|                                   | R  | 0.7100              | 0.7861              | 0.8622                     | 0.0033 |
|                                   | F1 | 0.6439              | 0.7129              | 0.7818                     |        |
| QuantConcept (35 SEHRs GPT-4o)    | P  | <b>0.5991</b>       | <b>0.6606</b>       | 0.7220                     |        |
|                                   | R  | <b>0.7230</b>       | <b>0.7971</b>       | 0.8713                     | 0.0032 |
|                                   | F1 | <b>0.6553</b>       | <b>0.7225</b>       | 0.7896                     |        |
| QuantConcept (35 pSEHRs + GPT-4o) | P  | 0.5928 <sup>†</sup> | 0.6596 <sup>†</sup> | 0.7264                     |        |
|                                   | R  | 0.7100              | 0.7900              | 0.8700                     | 0.0031 |
|                                   | F1 | 0.6462 <sup>†</sup> | 0.7189              | 0.7917                     |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.16: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Regarding the best results achieved for this particular STY, the higher performing sample 35 SEHRs GPT-4o. In Table 5.16, 35 SEHRs GPT-4o only performed worse for the type score across all three metrics relative to 35 pSEHRs + UMLS. With 300 EHRs, the best results, as with the Laboratory or Test Result STY, are spread along several samples. The 35 pSEHRs + GPT-4o setup has the higher P metrics. Also, in a broader view, we can see that the GPT-4o annotations achieved higher strict metrics with the 300 EHRs. The baseline outperformed three other metrics. With 100 EHRs, the results are more conclusive.

|                                   |    | Strict                    | Partial                   | Type                      | Loss   |
|-----------------------------------|----|---------------------------|---------------------------|---------------------------|--------|
| QuantConcept (baseline)           | P  | 0.5532                    | 0.6334                    | 0.7135                    | 0.0035 |
|                                   | R  | 0.6554                    | <b>0.7503</b>             | <b>0.8453</b>             |        |
|                                   | F1 | 0.6000                    | <b>0.6869</b>             | 0.7738                    |        |
| QuantConcept (35 pSEHRs + UMLS)   | P  | 0.5629                    | 0.6430                    | 0.7231                    | 0.0036 |
|                                   | R  | 0.6398 <sup>†</sup>       | 0.7308 <sup>†</sup>       | 0.8218 <sup>†</sup>       |        |
|                                   | F1 | 0.5989                    | 0.6841                    | 0.7693                    |        |
| QuantConcept (35 bSEHRs + UMLS)   | P  | 0.5653                    | 0.6491                    | 0.7329                    | 0.0036 |
|                                   | R  | 0.6359                    | 0.7302                    | 0.8244                    |        |
|                                   | F1 | 0.5985                    | 0.6873                    | <b>0.7760</b>             |        |
| QuantConcept (35 SEHRs GPT-4o)    | P  | 0.5610                    | 0.6353                    | 0.7095                    | 0.0036 |
|                                   | R  | <b>0.6580</b>             | 0.7451                    | 0.8322                    |        |
|                                   | F1 | <b>0.6056</b>             | 0.6858                    | 0.7660                    |        |
| QuantConcept (35 pSEHRs + GPT-4o) | P  | <b>0.5724<sup>†</sup></b> | <b>0.6543<sup>†</sup></b> | <b>0.7362<sup>†</sup></b> | 0.0035 |
|                                   | R  | 0.6320                    | 0.7224                    | 0.8127                    |        |
|                                   | F1 | 0.6007 <sup>†</sup>       | 0.6867 <sup>†</sup>       | 0.7726 <sup>†</sup>       |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.17: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

When compared to the GPT-4o annotations of the same 35 pSEHRs, the UMLS-based annotations surprisingly outperformed the GPT-4o annotations, although the difference was not large. With 300 EHRs, the GPT-4o annotations achieve a higher utility than the UMLS-based ones for the P and F1 metrics. With 100 EHRs, something peculiar happened.

Although it was expected that UMLS would perform worse for this STY, the fact is that it achieved strict scores of 0%. Looking at Table 5.6, we can see very few annotations for Quantitative Concept given by the UMLS-based annotations. Thus, we can say that it could be possible that these few annotations are affecting negatively the results. Also, the number of training steps might not be enough for the optimizer to achieve a set of trainable parameters capable of finding a lower loss and patterns to detect the entities.

However, even excluding the strict scores, the GPT-4o annotations of the pSEHRs outperform the UMLS-based ones with a big difference. Another detail to notice is that although the hypothesis made seems true for this STY regarding the annotations, the fact is that with SEHRs generated by GPT-4o, the utility increases more.

|                                   |    | Strict              | Partial             | Type                      | Loss   |
|-----------------------------------|----|---------------------|---------------------|---------------------------|--------|
| QuantConcept (baseline)           | P  | 0.3033              | 0.5082              | 0.7131                    |        |
|                                   | R  | 0.0962              | 0.1612              | 0.2263                    | 0.0060 |
|                                   | F1 | 0.1461              | 0.2448              | 0.3435                    |        |
| QuantConcept (35 pSEHRs + UMLS)   | P  | 0.0000              | 0.3548              | 0.7097                    |        |
|                                   | R  | 0.0000              | 0.0286              | 0.0572                    | 0.0066 |
|                                   | F1 | 0.0000              | 0.0529              | 0.1059                    |        |
| QuantConcept (35 bSEHRs + UMLS)   | P  | 0.2474              | 0.4691              | 0.6907                    |        |
|                                   | R  | 0.1248              | 0.2367              | 0.3485                    | 0.0054 |
|                                   | F1 | 0.1659              | 0.3146              | 0.4633                    |        |
| QuantConcept (35 SEHRs GPT-4o)    | P  | <b>0.4230</b>       | <b>0.5696</b>       | 0.7162                    |        |
|                                   | R  | <b>0.4109</b>       | <b>0.5533</b>       | <b>0.6957</b>             | 0.0045 |
|                                   | F1 | <b>0.4169</b>       | <b>0.5613</b>       | <b>0.7058</b>             |        |
| QuantConcept (35 pSEHRs + GPT-4o) | P  | 0.3302 <sup>†</sup> | 0.5408 <sup>†</sup> | <b>0.7514<sup>†</sup></b> |        |
|                                   | R  | 0.2263 <sup>†</sup> | 0.3706 <sup>†</sup> | 0.5150 <sup>†</sup>       | 0.0051 |
|                                   | F1 | 0.2685 <sup>†</sup> | 0.4398 <sup>†</sup> | 0.6111 <sup>†</sup>       |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.18: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

### 5.3.2.5 Diagnostic Procedure

The results using 600, 300, and 100 EHRs with different SEHR samples are shown in Tables 5.19, 5.20, and 5.21, respectively.

As for the best results achieved for this STY, a tendency can be seen as the number of EHRs decreases. With 600 EHRs, the best performances are split between the baseline and the sample 35 pSEHRs + UMLS overall. The baseline performance achieves the highest strict and partial F1, tied with the aforementioned sample. However, as the number of EHRs in the training corpus decreases, the sample 35 SEHRs GPT-4o achieves the highest performance considering the F1 measures. The baseline, however, continues to achieve the highest R metrics. In both 600 and 300 EHRs tests, the test losses were very similar. With 100 EHRs, the situation changes as now two fine-tuned BioBERTpt(all) models achieve a very poor performance, with the 35 bSEHRs + UMLS having scores of 0% across all metrics. The baseline also performs poorly. The 35 SEHRs GPT-4o continues to be the best-performing setup.

|                                     |    | Strict                    | Partial                   | Type                      | Loss   |
|-------------------------------------|----|---------------------------|---------------------------|---------------------------|--------|
| DiagnosticProc (baseline)           | P  | 0.6040                    | 0.6475                    | 0.6911                    | 0.0020 |
|                                     | R  | <b>0.7722</b>             | <b>0.8278</b>             | <b>0.8835</b>             |        |
|                                     | F1 | <b>0.6778</b>             | <b>0.7267</b>             | 0.7756                    |        |
| DiagnosticProc (35 pSEHRs + UMLS)   | P  | <b>0.6108<sup>†</sup></b> | <b>0.6605<sup>†</sup></b> | 0.7101 <sup>†</sup>       | 0.0021 |
|                                     | R  | 0.7468 <sup>†</sup>       | 0.8076 <sup>†</sup>       | 0.8684 <sup>†</sup>       |        |
|                                     | F1 | 0.6720 <sup>†</sup>       | <b>0.7267<sup>†</sup></b> | <b>0.7813<sup>†</sup></b> |        |
| DiagnosticProc (35 bSEHRs + UMLS)   | P  | 0.5824                    | 0.6294                    | 0.6765                    | 0.0020 |
|                                     | R  | 0.7519                    | 0.8127                    | 0.8734                    |        |
|                                     | F1 | 0.6564                    | 0.7094                    | 0.7624                    |        |
| DiagnosticProc (35 SEHRs GPT-4o)    | P  | 0.6013                    | 0.6561                    | <b>0.7110</b>             | 0.0019 |
|                                     | R  | 0.7215                    | 0.7873                    | 0.8532                    |        |
|                                     | F1 | 0.6559                    | 0.7158                    | 0.7756                    |        |
| DiagnosticProc (35 pSEHRs + GPT-4o) | P  | 0.5962                    | 0.6475                    | 0.6987                    | 0.0021 |
|                                     | R  | 0.7215                    | 0.7835                    | 0.8456                    |        |
|                                     | F1 | 0.6529                    | 0.7090                    | 0.7652                    |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.19: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

|                                     |    | Strict              | Partial             | Type                | Loss   |
|-------------------------------------|----|---------------------|---------------------|---------------------|--------|
| DiagnosticProc (baseline)           | P  | 0.5135              | 0.5987              | 0.6839              | 0.0029 |
|                                     | R  | <b>0.5797</b>       | <b>0.6759</b>       | <b>0.7722</b>       |        |
|                                     | F1 | 0.5446              | 0.6350              | <b>0.7253</b>       |        |
| DiagnosticProc (35 pSEHRs + UMLS)   | P  | 0.5130              | 0.6017              | 0.6903              | 0.0029 |
|                                     | R  | 0.5494              | 0.6443              | 0.7392 <sup>†</sup> |        |
|                                     | F1 | 0.5306              | 0.6222              | 0.7139 <sup>†</sup> |        |
| DiagnosticProc (35 bSEHRs + UMLS)   | P  | 0.4976              | 0.5952              | 0.6929              | 0.0030 |
|                                     | R  | 0.5291              | 0.6329              | 0.7367              |        |
|                                     | F1 | 0.5129              | 0.6135              | 0.7141              |        |
| DiagnosticProc (35 SEHRs GPT-4o)    | P  | <b>0.5622</b>       | <b>0.6343</b>       | <b>0.7065</b>       | 0.0030 |
|                                     | R  | 0.5722              | 0.6456              | 0.7190              |        |
|                                     | F1 | <b>0.5671</b>       | <b>0.6399</b>       | 0.7127              |        |
| DiagnosticProc (35 pSEHRs + GPT-4o) | P  | 0.5388 <sup>†</sup> | 0.6189 <sup>†</sup> | 0.6990 <sup>†</sup> | 0.0029 |
|                                     | R  | 0.5620 <sup>†</sup> | 0.6456 <sup>†</sup> | 0.7291              |        |
|                                     | F1 | 0.5502 <sup>†</sup> | 0.6320 <sup>†</sup> | 0.7138              |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.20: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

Regarding the results achieved that directly impact the hypothesis made, the outcome was not expected. The setup with the 35 pSEHRs annotated with UMLS achieved the highest utility for two of the three TRSTR variants. With 600 EHRs, the sample 35 pSEHRs + UMLS outperformed 35 pSEHRs + GPT-4o on all metrics across all scores with differences around 2%. With 100 EHRs, overall, the same outcome happened, this time with a lower score difference and having the GPT-4o annotations outperform the UMLS-based ones in one metric, Type P. In Table 5.20, however, the GPT-4o annotations outperformed the UMLS-based ones on all metrics except for type R and type F1. As with other STYs, the impact of the SEHRs generated by GPT-4o is visible compared to 35 pSEHRs + GPT-4o.

|                                     |    | Strict              | Partial                   | Type                      | Loss   |
|-------------------------------------|----|---------------------|---------------------------|---------------------------|--------|
| DiagnosticProc (baseline)           | P  | 0.0000              | 0.2500                    | 0.5000                    |        |
|                                     | R  | 0.0000              | 0.0025                    | 0.0051                    | 0.0060 |
|                                     | F1 | 0.0000              | 0.0050                    | 0.0100                    |        |
| DiagnosticProc (35 pSEHRs + UMLS)   | P  | 0.2919 <sup>†</sup> | <b>0.5189<sup>†</sup></b> | 0.7459                    |        |
|                                     | R  | 0.1367 <sup>†</sup> | 0.2430 <sup>†</sup>       | <b>0.3494<sup>†</sup></b> | 0.0049 |
|                                     | F1 | 0.1862 <sup>†</sup> | 0.3310 <sup>†</sup>       | <b>0.4759<sup>†</sup></b> |        |
| DiagnosticProc (35 bSEHRs + UMLS)   | P  | 0.0000              | 0.0000                    | 0.0000                    |        |
|                                     | R  | 0.0000              | 0.0000                    | 0.0000                    | 0.0067 |
|                                     | F1 | 0.0000              | 0.0000                    | 0.0000                    |        |
| DiagnosticProc (35 SEHRs GPT-4o)    | P  | <b>0.3089</b>       | 0.5157                    | 0.7225                    |        |
|                                     | R  | <b>0.1494</b>       | <b>0.2494</b>             | <b>0.3494</b>             | 0.0049 |
|                                     | F1 | <b>0.2014</b>       | <b>0.3362</b>             | 0.4710                    |        |
| DiagnosticProc (35 pSEHRs + GPT-4o) | P  | 0.2658              | 0.5158                    | <b>0.7658<sup>†</sup></b> |        |
|                                     | R  | 0.1063              | 0.2063                    | 0.3063                    | 0.0050 |
|                                     | F1 | 0.1519              | 0.2948                    | 0.4376                    |        |

<sup>†</sup> Indicates higher performance when comparing 35 pSEHRs + UMLS with 35 pSEHRs + GPT-4o considering the hypothesis.

Table 5.21: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 pSEHRs annotated using the UMLS-based weak annotation tool

## 5.4 Further SDG adjustments and final results for SemClinBr

Considering the results achieved in Section 5.3, despite no statistical inference test being made, they seemed to enlighten the answer to the hypothesis made. The results showed, overall, that indeed the annotations were impacting the SEHR utility more than the texts generated using GPT-3.5.

Promising results were also achieved when using GPT-4o with ChatGPT to generate and annotate SEHRs. GPT-4o annotations indeed seemed to have higher utility than the UMLS-based weak annotations. The synthetic texts generated by GPT-4o also seemed to have higher utility than those generated by GPT-3.5, at least for certain STYs.

This section will start by focusing on the preparatory work made before using the OpenAI API to streamline the SDG process. This section ends with the final results achieved with the SD to the SemClinBr benchmark, both realism and utility.

Section 5.4.1 will show the SDG results achieved after doing more prompt engineering to the most updated and performing prompt, which was used to generate and annotate 35 SEHRs using GPT-4o. Section 5.4.2 goes into the details of the two SEHR datasets created using two versions of GPT-4o. These results consist of a quantitative evaluation of privacy disclosure and resemblance. Section 5.4.3 shows the NER results achieved for both the prompt engineering improvements made, and also for the two SEHR datasets developed, thus achieving the final utility results for SemClinBr.

### 5.4.1 SEHRs document and annotation generation adjustments

The first prompt achieved, after including a better selection of external files to feed the RAG tool, File Search, as well as other details, can be seen below.

**[System instructions]** ""You are an advanced AI language model that generates and annotates synthetic health reports in Portuguese. Your task is to generate high-quality health reports and provide accurate Named Entity Recognition annotations based on the provided definitions and examples.

### Guidelines:

1. Language: Generate text in Portuguese.
2. Health Report Context: Ensure the text reflects typical health report scenarios. Follow the structure of the reports uploaded to the system instructions.
3. Annotations: Annotate entities accurately using the provided definitions and examples. Follow the format used in the file examples uploaded to the system instructions.

### Annotation Tag Definitions:

1. Disease or Syndrome: A condition that affects the body or mind, with specific symptoms and signs.
2. Sign or Symptom: Signs are objective evidence of disease perceptible to the examining physician, while symptoms are the patient's subjective experiences.
3. Quantitative Concept: A concept that represents measurable quantities. These could include physical quantities, such as length or mass.
4. Laboratory or Test Result: The outcome of a laboratory test or diagnostic procedure. This result provides information about the extent of disease, condition, or abnormality.
5. Diagnostic Procedure: A medical process to determine the presence, extent, or cause of a disease or condition. Diagnostic procedures include laboratory tests, biopsies, and other methods used to diagnose medical conditions. ""

**[User instructions]** ""Considering the system instructions given and the files uploaded to it, generate a complete, extensive, and high-quality electronic health report, with around 40 entities for tag annotation across all annotation tags, for a patient with the following condition: "CONCEPT". Generate annotations according to the annotation tag definitions, while considering all the tags mentioned in it. Also, pay close attention to how the tags were used in the uploaded file examples. Use several abbreviations just like in the file examples uploaded. Generate an XML file format like the text files I uploaded. Your output/response should only have the generated file. ""

**Prompt GPT-4o adjusted:** GPT-4o prompt adjusted by translating the user prompt to English and adding a system prompt, where CONCEPT is a disease or condition retrieved from UMLS in Portuguese.

Perhaps the change that stimulated the time effort to improve Prompt 35 SEHRs GPT-4o was the external EHR files uploaded to ChatGPT via File Search. In Prompt 35 SEHRs GPT-4o, the only criteria for the external files were that they needed to be part of the 600 training EHRs. What

this means is that a total of 4 files were chosen at random from the training corpus.

Therefore, in the adjusted prompt shown above, the four files uploaded were replaced by different training files. As for the criterion used for their selection, a non-extensive search was made to find training EHRs with a larger amount of annotations for each STYs relative to the files used in Prompt 35 SEHRs GPT-4o. That comparison can be seen in Figure 5.3. Besides that, special attention was given to STYs which did not see much NER improvement using the SEHRs generated and annotated in Section 5.3.

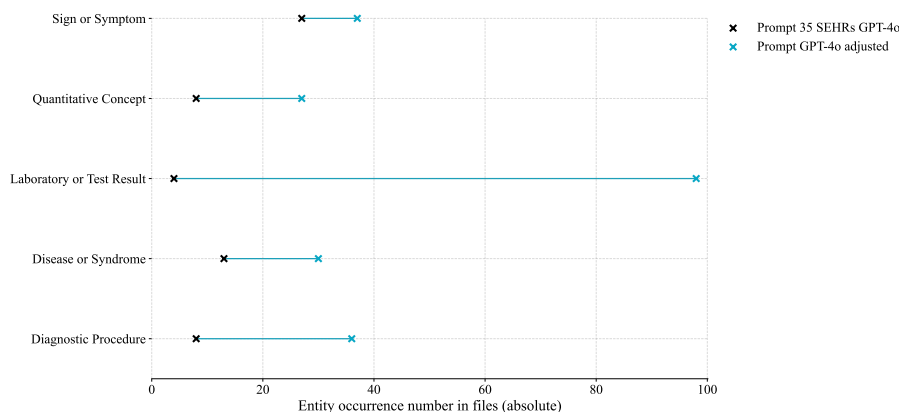


Figure 5.3: Dumbbell plot of Entity STY occurrences in the 4 files selected for each prompt

One final adjustment was made to the four EHR files used in the Prompt GPT-4o adjusted. All annotations regarding entities whose STYs were not relevant were removed from the files. We made this removal for two main reasons: to prevent the LLM from being more confused, and to reduce the prompt cost by decreasing the number of input tokens.

A second variant of such prompt was created to check if it was possible to have SEHRs with higher utility. For that, a Prompt GPT-4o adjusted was modified to exclude the CONCEPT component. This prompt will be named from now on Prompt GPT-4o adjusted v2. With this modification, it was expected the BLEU scores would increase in this modified version relative to the Prompt GPT-4o adjusted.

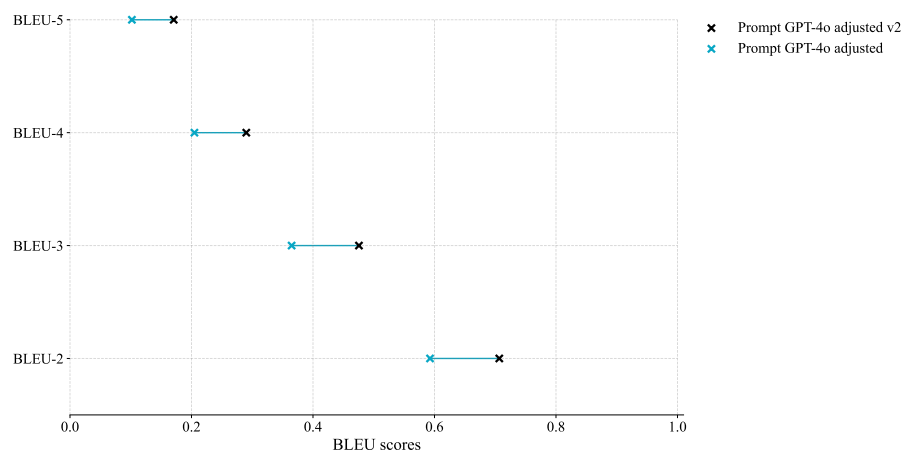


Figure 5.4: Dumbbell plot of BLEU scores shows a decrease in privacy disclosure by opting by a prompt which comprises a randomness component, CONCEPT

The BLEU scores were measured considering as references all 600 EHRs used for training and as candidates all 35 SEHRs for each of the samples. The results can be seen in Figure 5.4. We can see that removing the CONCEPT component increases privacy disclosure risks.

Regarding the resemblance between the synthetic sentences and the sentences retrieved from the 600 training EHRs, scatterplots comparing side-to-side the 2-dimensional latent space of their respective sentence embeddings can be seen in Figures 5.5 and 5.6. Figure 5.5 used PCA as a dimensionality reduction technique, while in Figure 5.6 t-SNE was applied. BioBERTpt(all) was used to compute the sentence embeddings. KL divergences between the distribution of real and synthetic latent components were also computed.

Looking at the 2-dimensional mapping in Figure 5.5, using PCA shows a similar tendency across both prompts in terms of sentence-embedded meaning. That being said, the discrepancy in terms of the number of sentences between the real and synthetic EHRs is considerable. This is because the number of real EHRs is 600 compared to the 35 SEHRs generated using those two prompts. Still, there seems to exist a larger similarity, although quite small, between sentences looking at the scatterplot on the right, meaning Prompt GPT-4o adjusted. In Figure 5.6, despite from a mere subjective perspective, the scatterplot on the right appears to have a more similar sentence embedding distribution.

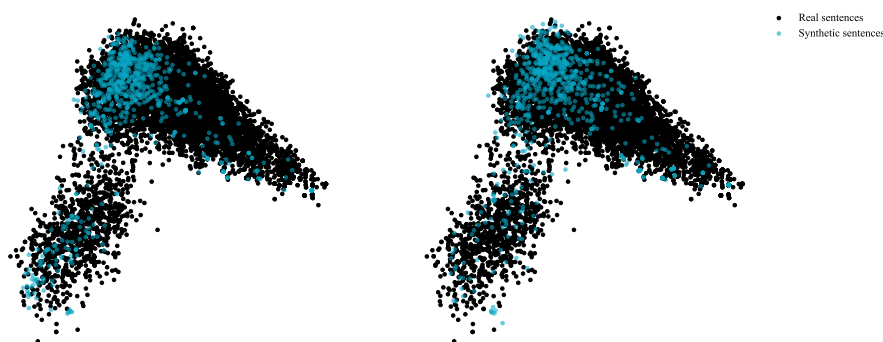


Figure 5.5: Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using PCA. The scatterplot on the left corresponds to Prompt GPT-4o adjusted v2 (KL divergence equal to 0.30574), while the one on the right corresponds to Prompt GPT-4o adjusted (KL divergence equal to 0.18830).

However, just like in Figure 5.5, the t-SNE mapping also shows synthetic sentences being more dispersed when using the Prompt GPT-4o adjusted, plot on the right. Just like with the BLEU scores, considering the only difference between prompts is the usage of the CONCEPT component, the prompt with it generated SEHRs with more dispersed sentence embeddings. Regarding KL divergences, across PCA and t-SNE the same analysis can be made. The entropy between distributions shown in Figures 5.5 and 5.6 is lower with Prompt GPT-4o adjusted. Thus, the distribution between the training and synthetic sentence embedding’s latent variables is similar in the SEHRs generated using Prompt GPT-4o adjusted.

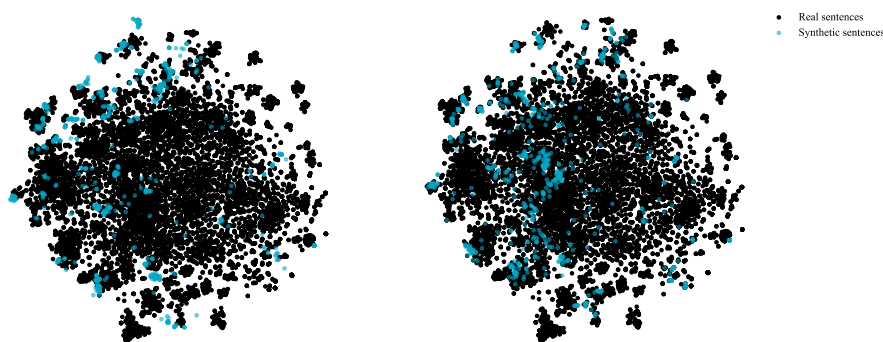


Figure 5.6: Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using t-SNE. The scatterplot on the left corresponds to Prompt GPT-4o adjusted v2 (KL divergence equals to 33.49310), while the one on the right corresponds to Prompt GPT-4o adjusted (KL divergence equals to 23.28585).

Prompt GPT-4o adjusted was chosen, considering the BLEU scores, resemblance, and the NER results, the last ones detailed in Section 5.4.3. Minor tests were made with the hyperparameters of GPT-4o. Due to the cost and budget established, these minor tests were kept to 3 outputs

per combination of temperature and top-p threshold.

With high and low values for both hyperparameters, gpt-4o-2024-05-13 and gpt-4o-2024-08-06 would give at least one SEHR with very few annotations. It was found that with a temperature and a top-p threshold of 0.5 both GPT-4o models gave a more constant number of annotations per SEHR generated. Notice that in the user's prompt, it was added the number of desired annotations per SEHR, 40 annotations. This was done also to try to have a SDG pipeline that could generate a more constant number of annotations.

### 5.4.2 Final SEHR datasets for SemClinBr augmentation

Having decided on the final prompt, Prompt GPT-4o adjusted, the final step before measuring the last NER results was to generate a larger amount of SEHRs.

A total of 600 SEHRs were generated, 300 SEHRs per GPT-4o snapshot model. In absolute terms, gpt-4o-2024-05-13 generated a total of 4,658 sentences, while gpt-4o-2024-08-06 generated 7,263 sentences. Relatively speaking, the latter generated 61% of all sentences. This also means that for the same amount of SEHRs, gpt-4o-2024-08-06 generated 56% more sentences relative to gpt-4o-2024-05-13.

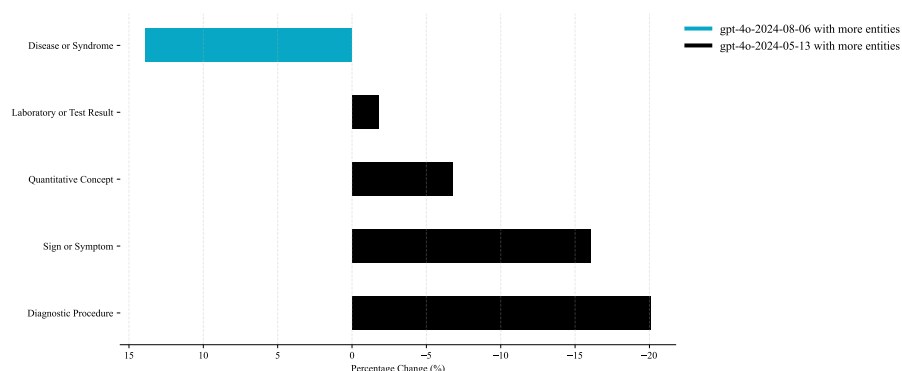


Figure 5.7: Horizontal barplot with percentage variation of number of entities per STY of gpt-4o-2024-08-06 relative to gpt-4o-2024-05-13. Color encoding helps the perception of which SEHR dataset has more entities per STY.

The absolute frequencies of entities per STY can be seen in Table 5.22, while in Figure 5.7 the percentage changes relative to gpt-4o-2024-05-13 is presented. Despite the argument made before regarding the choice of the temperature and top-p threshold for the GPT-4o snapshot models, still, a considerable change in the total number of entities, as well as per STY, is noticeable. gpt-4o-2024-05-13 generated a total of 10,549 entities, while gpt-4o-2024-08-06 generated 9,605 entities.

It was interesting to see that different GPT-4o models returned a noticeable change in the number of entities per STY generated, although one could argue that the CONCEPT component may have also impacted that result. Still, Figure 5.7 shows that only one STY, Disease or Syndrome, had a higher number of entities annotated by gpt-4o-2024-08-06 relative to gpt-4o-2024-05-13. Following such comparison, gpt-4o-2024-08-06 generated 13.9% more entities for Disease

or Syndrome, while for the other STYs, the variations were -1.8%, -6.8%, -16.0%, and -20.1%, respectively.

| Semantic Type             | gpt-4o-2024-05-13 | gpt-4o-2024-08-06 |
|---------------------------|-------------------|-------------------|
| Disease or Syndrome       | 965               | 1,099             |
| Laboratory or Test Result | 2,226             | 2,187             |
| Quantitative Concept      | 2,186             | 2,038             |
| Sign or Symptom           | 3,637             | 3,054             |
| Diagnostic Procedure      | 1,535             | 1,227             |

Table 5.22: Number of occurrences of each STY across both SEHR datasets generated using gpt-4o-2024-05-13 and gpt-4o-2024-08-06

After checking these basic descriptive statistics, it was relevant to check if there was one model in particular that, despite all the SDG pipeline tuning made thus far, had a higher risk of privacy disclosure. For that, like in previous sections, the BLEU scores were compared using a dumbbell plot, which can be seen in Figure 5.8. Such results, despite being very similar, show slightly higher BLEU scores for the SEHRs generated by gpt-4o-2024-08-06.

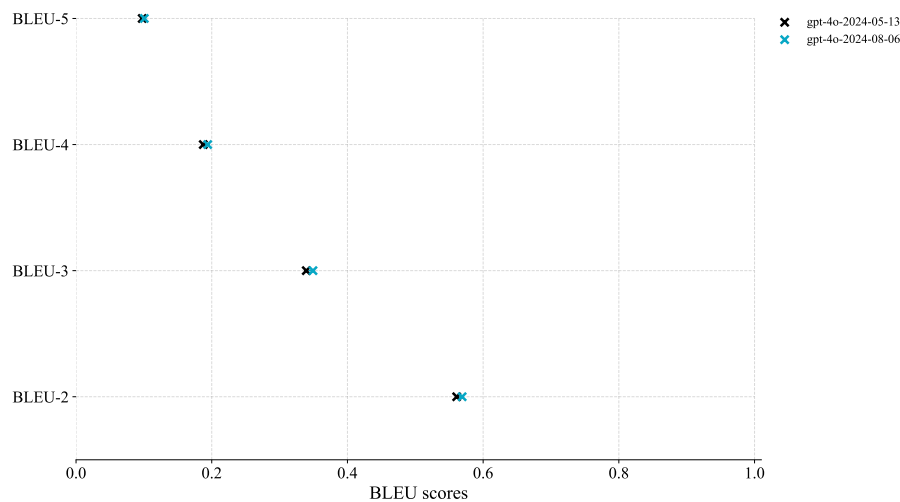


Figure 5.8: Dumbbell plot of BLEU scores shows very similar values from the BLEU scores for bigrams until 5-grams. These results were achieved considering as references the 600 training EHRs and as candidates the SEHRs generated either by gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

However, just like with the BLEU scores, the following results regarding resemblance are the final results obtained for the 2 SEHR datasets developed. The number of sentences compared with the previous experience has increase significantly. The application of PCA and t-SNE to the sentence embeddings of the training and synthetic sentences, for each SEHR dataset, can be seen in Figures 5.9 and 5.10, respectively.

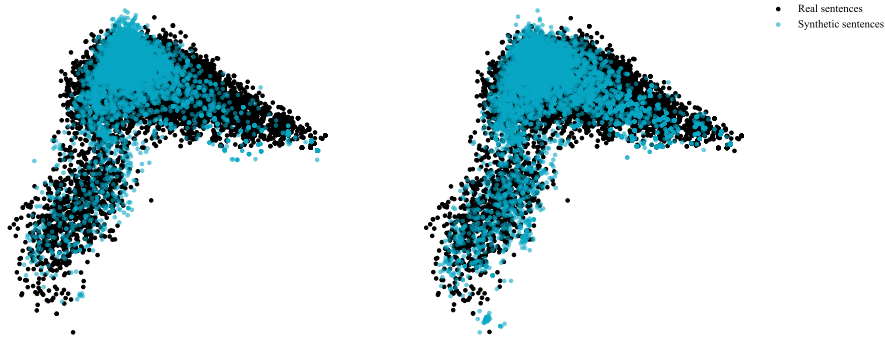


Figure 5.9: Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using PCA. The scatterplot on the left corresponds to gpt-4o-2024-05-13 (KL divergence equal to 0.27300), while the one on the right corresponds to gpt-4o-2024-08-06 (KL divergence equal to 0.09637).



Figure 5.10: Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using t-SNE. The scatterplot on the left corresponds to gpt-4o-2024-05-13 (KL divergence equal to 15.59174), while the one on the right corresponds to gpt-4o-2024-08-06 (KL divergence equal to 10.43920).

Looking at the results, we can see that gpt-4o-2024-08-06 shows more dispersed sentence embeddings across the 2-dimensional latent spaces, either computed using PCA or t-SNE. Regarding KL divergence, between PCA and t-SNE, the entropy between distributions shown in Figures 5.9 and 5.10 is lower with gpt-4o-2024-08-06. Meaning, the distribution between the training and synthetic sentence embedding's latent variables is similar in the SEHRs generated with gpt-4o-2024-08-06.

However, besides measuring KL divergence, we decided to check if there were statistically significant differences between the cosine similarities between real and synthetic sentence embeddings. The main idea was to check if the conclusions with KL divergence and cosine similarity were the same. The two independent samples were the cosine similarities which came from the SEHR's sentence embeddings for each GPT-4o snapshot model.

There was just a problem, the sample size. The sentence embeddings relative to gpt-4o-2024-05-13 and gpt-4o-2024-08-06 resulted in a total of 46,160,780 and 71,976,330 similarity scores in the similarity score matrix. There is a known problem with large samples which is associated with the Law of Large Numbers (LLN). As the sample size increases, the LLN states that the sample mean converges to the population mean. However, when that happens, even small changes in the data can lead to statistical significance considering that the variance tends to decrease as the sample size increases. In other words, statistical significance is affected by the sample size.

So, the approach to test for statistically significant differences between the cosine similarities was to check, instead, if the difference between the means is statistically significant using bootstrap sampling. A total of 1,000 means for each group were computed by retrieving 500 similarity scores with replacement from each independent sample. The boxplots for both samples of means can be seen in Figure 5.11. Just by looking at the box's lack of intersection, one can say that there is a big chance the differences between the two samples will be statistically significant.

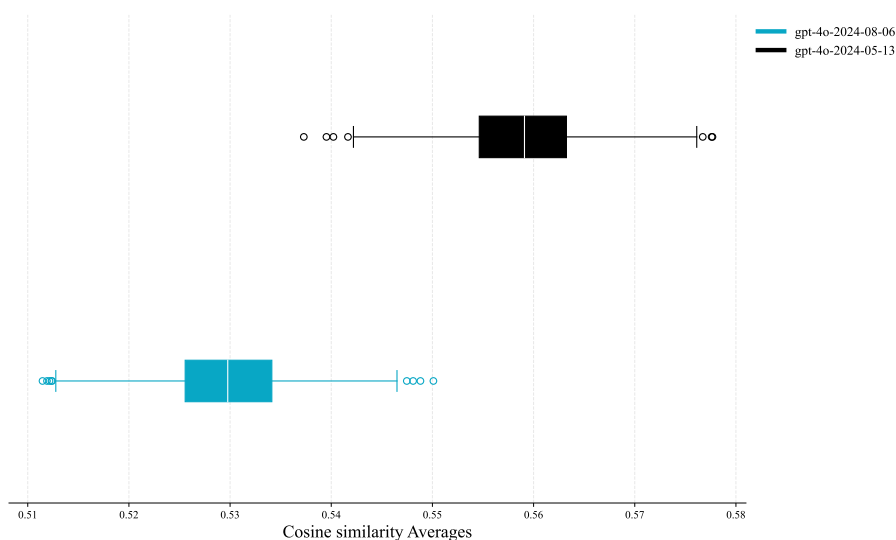


Figure 5.11: Boxplot of the two samples of averages for the cosine similarity scores between real and synthetic sentences for gpt-4o-2024-05-13 and gpt-4o-2024-08-06.

To test the normality of both samples, the Shapiro-Wilk test was used, achieving p-values of 0.527 and 0.492 for gpt-4o-2024-05-13 and gpt-4o-2024-08-06, respectively. The Levene test, to check for equal variances between these two groups, achieved a p-value of 0.581. All these results drove to not reject the null hypothesis of both tests, thus showing that both samples follow a normal distribution and their variances are equal. Thus, a two-sided parametric independent t-test was made, achieving a t-statistic of 101.34 and a p-value lower than the significance level of 0.05. The confidence interval for such mean differences was (0.0285, 0.0297).

A one-sided t-test is not required to understand that the cosine similarities of gpt-4o-2024-05-13 are statistically higher than the ones from gpt-4o-2024-08-06. However, as expected, considering as null hypothesis of a one-sided t-test that the gpt-4o-2024-05-13's similarity scores

were statistically higher than those of gpt-4o-2024-08-06, a p-value of 1.0 was achieved, thus not rejecting the null hypothesis above.

These significant results, however, give a different analysis relative to the KL divergences.

### 5.4.3 Participant extraction

To choose between Prompt GPT-4o adjusted and Prompt GPT-4o adjusted v2, the 35 SEHRs generated using GPT-4o on ChatGPT were used to augment the TRSTR scheme with 100 EHRs. This was the only scheme used to measure the utility each prompt was capable of generating. Such prompt details, as well as privacy disclosure and resemblance results, can be seen in Section 5.4.1.

Regarding the NER results comparing both prompts with the respective baseline can be seen in Tables 5.23, 5.24, 5.25, 5.26, and 5.27 for Sign or Symptom, Disease or Syndrome, Laboratory or Test Result, Quantitative Concept, and Diagnostic procedure, respectively. The results regarding the strict F1 scores can be seen in Figure 5.12, considering also 35 SEHRs GPT-4o and 35 pSEHRs + GPT-4o for comparison.

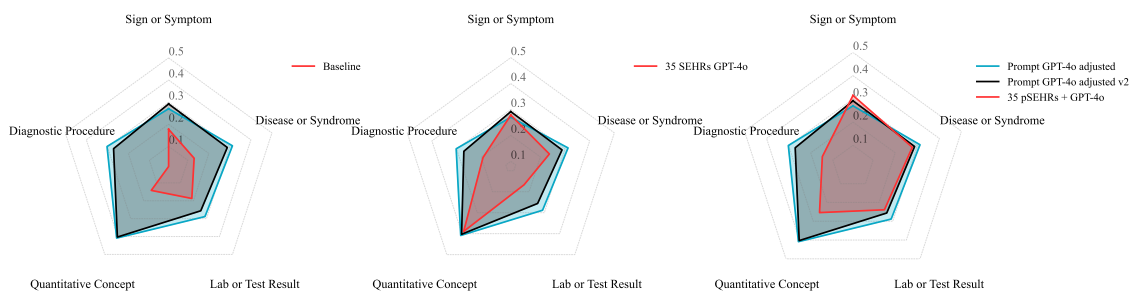


Figure 5.12: Radial plot of strict F1 scores across all STYs relative to Prompt GPT-4o adjusted, Prompt GPT-4o adjusted v2, and the respective baseline. Such a comparison can be seen on the left radial plot. On the middle radial plot both Prompt GPT-4o adjusted and Prompt GPT-4o adjusted v2 are compared against the 35 SEHRs GPT-4o performance on 100 EHRs. On the right, such a comparison is made against the 35 pSEHRs + GPT-4o.

Looking at the strict F1 performances across all five STYs it is noticeable that in four out of five Prompt GPT-4o adjusted outperformed Prompt GPT-4o adjusted v2. The baseline was always outperformed by either of them, overall.

An interesting comparison to make was to check how the final prompt engineering adjustments contributed to improve the NER performance and utility. For that, the baseline performance displayed in the color red, in Figure 5.12, was replaced by the 35 SEHRs GPT-4o and 35 pSEHRs + GPT-4o performances on the 100 EHRs TRSTR scheme variant, showcased in the middle and right radial plot, respectively. Regarding such comparison, both adjusted GPT-4o prompts were able to generate SEHRs capable of outperforming 35 SEHRs GPT-4o and 35 pSEHRs + GPT-4o on the vast majority of STYs, with some exceptions on the Sign or Symptom STY.

Regarding the visualization technique selected and the data encoding, it is noticeable a perception illusion, specifically due to the radial plot in the middle. Notice how it seems that the Prompt

GPT-4o adjusted and Prompt GPT-4o adjusted v2 filled areas are smaller in the middle radial plot relative to those same filled areas on the left and right radial plots. This seems to be happening because the 35 SEHRs GPT-4o strict F1 scores for Diagnostic Procedure and Lab or Test Result are lower, creating a shape with a lower width.

It could be argued that the visualization technique should have been replaced by another. We did not do that because our brains are used to associate radial plots with performance comparison. This is the case, for example, in video games. So, we decided it was better to keep the radial plot, but mentioning such detail. We could have also swapped the axis order, but the order in which we've been analysing the STYs NER results is the same as in the radial plots, clockwise. Despite not being intended, such perception illusion was kept to emphasize perception illusions can affect the audience's perception, for better or worse.

|                                           |    | Strict        | Partial       | Type          | Loss          |
|-------------------------------------------|----|---------------|---------------|---------------|---------------|
| SignOrSymptom (baseline)                  | P  | 0.2180        | 0.4087        | 0.5995        | 0.0131        |
|                                           | R  | 0.1600        | 0.3000        | 0.4400        |               |
|                                           | F1 | 0.1845        | 0.3460        | 0.5075        |               |
| SignOrSymptom (prompt GPT-4o adjusted)    | P  | 0.2551        | 0.3811        | 0.5070        | <b>0.0110</b> |
|                                           | R  | <b>0.3260</b> | <b>0.4870</b> | <b>0.6480</b> |               |
|                                           | F1 | 0.2862        | 0.4276        | 0.5689        |               |
| SignOrSymptom (prompt GPT-4o adjusted v2) | P  | <b>0.3278</b> | <b>0.4649</b> | <b>0.6020</b> | 0.0121        |
|                                           | R  | 0.2940        | 0.4170        | 0.5400        |               |
|                                           | F1 | <b>0.3100</b> | <b>0.4396</b> | <b>0.5693</b> |               |

Table 5.23: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2.

|                                               |    | Strict        | Partial       | Type          | Loss          |
|-----------------------------------------------|----|---------------|---------------|---------------|---------------|
| DiseaseOrSyndrome (baseline)                  | P  | 0.3151        | 0.5034        | <b>0.6918</b> | 0.0081        |
|                                               | R  | 0.0842        | 0.1346        | 0.1850        |               |
|                                               | F1 | 0.1329        | 0.2124        | 0.2919        |               |
| DiseaseOrSyndrome (prompt GPT-4o adjusted)    | P  | 0.3588        | 0.4809        | 0.6030        | <b>0.0068</b> |
|                                               | R  | <b>0.3095</b> | <b>0.4148</b> | <b>0.5201</b> |               |
|                                               | F1 | <b>0.3324</b> | <b>0.4454</b> | <b>0.5585</b> |               |
| DiseaseOrSyndrome (prompt GPT-4o adjusted v2) | P  | <b>0.3708</b> | <b>0.5052</b> | 0.6397        | 0.0070        |
|                                               | R  | 0.2601        | 0.3544        | 0.4487        |               |
|                                               | F1 | 0.3057        | 0.4166        | 0.5274        |               |

Table 5.24: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2.

|                                             |    | Strict        | Partial       | Type          | Loss          |
|---------------------------------------------|----|---------------|---------------|---------------|---------------|
| LabOrTestResult (baseline)                  | P  | 0.2220        | 0.4260        | 0.6300        | 0.0093        |
|                                             | R  | 0.1747        | 0.3353        | 0.4958        |               |
|                                             | F1 | 0.1955        | 0.3752        | 0.5549        |               |
| LabOrTestResult (prompt GPT-4o adjusted)    | P  | <b>0.3048</b> | <b>0.4588</b> | <b>0.6129</b> | 0.0092        |
|                                             | R  | <b>0.3078</b> | 0.4634        | 0.6190        |               |
|                                             | F1 | <b>0.3063</b> | <b>0.4611</b> | 0.6159        |               |
| LabOrTestResult (prompt GPT-4o adjusted v2) | P  | 0.2440        | 0.4013        | 0.5587        | <b>0.0086</b> |
|                                             | R  | 0.3045        | <b>0.5008</b> | <b>0.6972</b> |               |
|                                             | F1 | 0.2709        | 0.4456        | <b>0.6203</b> |               |

Table 5.25: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2.

|                                          |    | Strict        | Partial       | Type          | Loss          |
|------------------------------------------|----|---------------|---------------|---------------|---------------|
| QuantConcept (baseline)                  | P  | 0.3033        | 0.5082        | 0.7131        |               |
|                                          | R  | 0.0962        | 0.1612        | 0.2263        | 0.0060        |
|                                          | F1 | 0.1461        | 0.2448        | 0.3435        |               |
| QuantConcept (prompt GPT-4o adjusted)    | P  | 0.4327        | 0.5635        | 0.6942        |               |
|                                          | R  | <b>0.4434</b> | <b>0.5774</b> | <b>0.7113</b> | <b>0.0046</b> |
|                                          | F1 | <b>0.4380</b> | <b>0.5703</b> | <b>0.7026</b> |               |
| QuantConcept (prompt GPT-4o adjusted v2) | P  | <b>0.5070</b> | <b>0.6439</b> | <b>0.7807</b> |               |
|                                          | R  | 0.3758        | 0.4772        | 0.5787        | 0.0048        |
|                                          | F1 | 0.4317        | 0.5482        | 0.6647        |               |

Table 5.26: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2.

|                                            |    | Strict        | Partial       | Type          | Loss          |
|--------------------------------------------|----|---------------|---------------|---------------|---------------|
| DiagnosticProc (baseline)                  | P  | 0.0000        | 0.2500        | 0.5000        |               |
|                                            | R  | 0.0000        | 0.0025        | 0.0051        | 0.0060        |
|                                            | F1 | 0.0000        | 0.0050        | 0.0100        |               |
| DiagnosticProc (Prompt GPT-4o adjusted)    | P  | <b>0.3608</b> | <b>0.5079</b> | 0.6551        |               |
|                                            | R  | <b>0.2886</b> | <b>0.4063</b> | <b>0.5241</b> | <b>0.0045</b> |
|                                            | F1 | <b>0.3207</b> | <b>0.4515</b> | <b>0.5823</b> |               |
| DiagnosticProc (Prompt GPT-4o adjusted v2) | P  | 0.3440        | 0.5018        | <b>0.6596</b> |               |
|                                            | R  | 0.2456        | 0.3582        | 0.4709        | 0.0047        |
|                                            | F1 | 0.2866        | 0.4180        | 0.5495        |               |

Table 5.27: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 35 SEHRs generated either using Prompt GPT-4o adjusted or Prompt GPT-4o adjusted v2.

Despite Prompt GPT-4o adjusted v2 having for several STYs the highest P metrics, the R metrics for the same scores don't follow such relatively high precision scores. On the other hand, Prompt GPT-4o adjusted can achieve a balance between P and R across all scores, overall. Hence being able to achieve higher F1 scores. Such differences for the strict P and R can be seen in Figure 5.13.

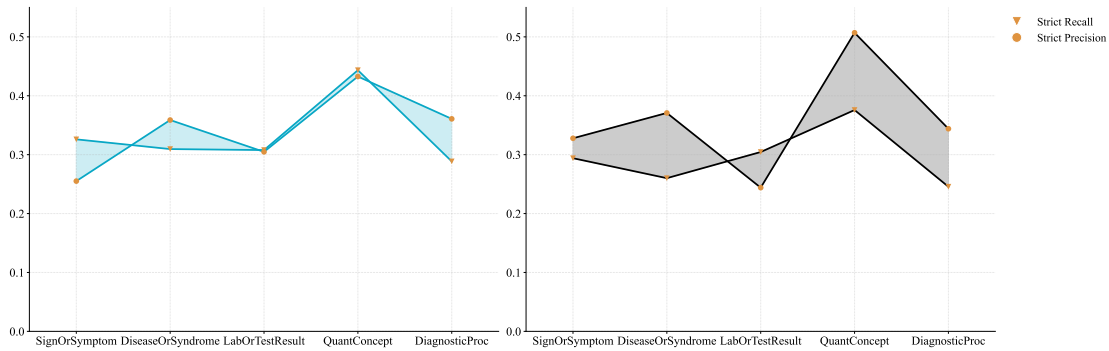


Figure 5.13: Parallel coordinates plot for the differences between strict P and R for Prompt GPT-4o adjusted (left) and Prompt GPT-4o adjusted v2 (right). Prompt GPT-4o adjusted is able to keep strict P and R performance closer than Prompt GPT-4o adjusted v2.

After considering these results, as well as the privacy disclosure and resemblance results achieved in Section 5.4.1, and as mentioned at the beginning of Section 5.4.2, the prompt chosen to generate the final SEHRs to measure utility was Prompt GPT-4o adjusted.

With the final prompt achieved, a total of 600 SEHRs were generated, 300 using a different GPT-4o snapshot model, gpt-4o-2024-05-13 and gpt-4o-2024-08-06. The final results for Sign or Symptom, Disease or Syndrome, Laboratory or Test Result, Quantitative Concept, and Diagnostic Procedure, can be seen in the tables in Sections 5.4.3.1, 5.4.3.2, 5.4.3.3, 5.4.3.4, and 5.4.3.5, respectively, together with a more tailored exposition of the results for each STY. On the other hand, the aggregated results for the strict F1 metric, across all STYs and TRSTR scheme variants, compared to their respective baselines, can be seen in the three radial plots in Figure 5.14.

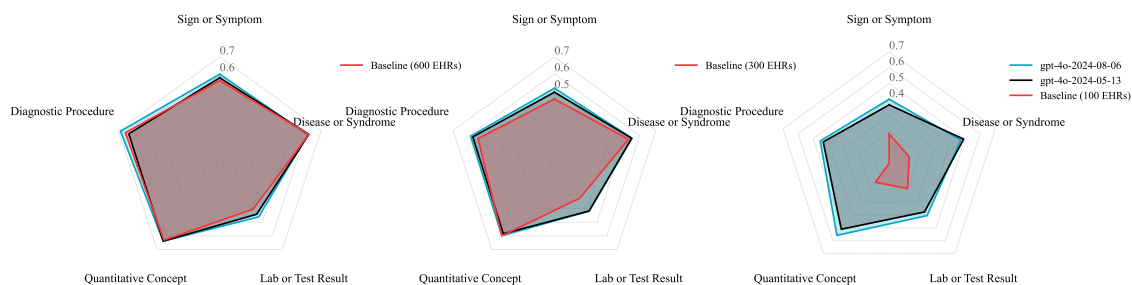


Figure 5.14: Radial plot of strict F1 scores across all STYs relative to gpt-4o-2024-05-13, gpt-4o-2024-08-06, and the respective baselines with either 600 EHRs (right), 300 EHRs (middle), or 100 EHRs (left). Utility decreases as the number of real EHRs available for training increases. Still, with 600 real EHRs, gpt-4o-2024-08-06 continues to deliver higher performance for strict F1 relative to the baseline, overall. gpt-4o-2024-08-06 achieves the highest utility.

The scale on the radial plots was kept the same so the strict F1 improvement's evolution could be better perceived across different baselines. Overall, it is noticeable that utility decreases as the number of real EHRs available for training also increases. This happens because the SEHR's added value decreases as more real EHRs are available for training. With this NER benchmark, 100 real EHRs is not enough to achieve a relatively high performance. In that instance, the SEHRs improve the performance with the largest margin of all three TRSTR schemes.

Comparing the radial plots for the TRSTR schemes with 100 and 300 real EHRs, notice how the baseline from the latter performs closely with the SEHRs generated by gpt-4o-2024-05-13 and gpt-4o-2024-08-06 which augmented the 100 EHRs from the former. The only STY where the difference between those performances seems to be larger is for Diagnostic Procedures.

For example, let's consider the gpt-4o-2024-08-06 performance on the TRSTR scheme with 100 EHRs relatively to the baseline with 300 EHRs. The strict F1 performance of the former for Sign or Symptom, Disease or Syndrome, Laboratory or Test Result, and Quantitative Concept was 0.4026, 0.4720, 0.4051, and 0.5582, respectively. On the other hand, the latter's performance was 0.4505, 0.5305, 0.3247, and 0.6000, respectively. Although it is true that the baseline with 300 EHRs performed better overall on this metric, relatively to gpt-4o-2024-08-06 with 100 EHRs, the performance is somewhat close, except for Disease or Syndrome. Also, gpt-4o-2024-08-06 with 100 EHRs outperformed the mentioned baseline on Laboratory or Test Result. These results can be seen in Tables 5.29 and 5.30 for Sign or Symptom, Tables 5.32 and 5.33 for Disease or Syndrome,

Tables 5.35 and 5.36 for Laboratory or Test Result, and Tables 5.38 and 5.39 for Quantitative Concept.

#### 5.4.3.1 Sign or Symptom

This section shows the results of BioBERTpt(all) fine-tuned for the Sign or Symptom STY with 300 SEHRs either generated by gpt-4o-2024-05-13 or gpt-4o-2024-08-06, across the three TRSTR schemes considered for this NER benchmark, with 600, 300, and 100 EHRs. Those results can be seen in Tables 5.28, 5.29, and 5.30, respectively.

|                                   |    | Strict        | Partial       | Type          | Loss          |
|-----------------------------------|----|---------------|---------------|---------------|---------------|
| SignOrSymptom (baseline)          | P  | 0.4888        | 0.5644        | <b>0.6400</b> |               |
|                                   | R  | 0.6530        | 0.7540        | <b>0.8550</b> | <b>0.0074</b> |
|                                   | F1 | 0.5591        | 0.6455        | <b>0.7320</b> |               |
| SignOrSymptom (gpt-4o-2024-08-06) | P  | <b>0.5212</b> | <b>0.5759</b> | 0.6307        |               |
|                                   | R  | <b>0.7000</b> | <b>0.7735</b> | 0.8470        | <b>0.0074</b> |
|                                   | F1 | <b>0.5975</b> | <b>0.6603</b> | 0.7230        |               |
| SignOrSymptom (gpt-4o-2024-05-13) | P  | 0.4985        | 0.5562        | 0.6139        |               |
|                                   | R  | 0.6830        | 0.7620        | 0.8410        | 0.0080        |
|                                   | F1 | 0.5764        | 0.6430        | 0.7097        |               |

Table 5.28: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

For this STY, across all TRSTR schemes, both SEHRs generated by gpt-4o-2024-05-13 and gpt-4o-2024-08-06 achieved a positive utility as they, overall, outperformed the baseline across all metrics and scores. As expected, however, such utility increases as the number of real EHRs available decreases, as well as the number of metrics with higher performance.

|                                   |    | Strict        | Partial       | Type          | Loss          |
|-----------------------------------|----|---------------|---------------|---------------|---------------|
| SignOrSymptom (baseline)          | P  | 0.3929        | 0.4900        | 0.5871        |               |
|                                   | R  | 0.5280        | 0.6585        | 0.7890        | 0.0091        |
|                                   | F1 | 0.4505        | 0.5619        | 0.6732        |               |
| SignOrSymptom (gpt-4o-2024-08-06) | P  | <b>0.4528</b> | <b>0.5285</b> | <b>0.6043</b> |               |
|                                   | R  | 0.5950        | 0.6945        | 0.7940        | <b>0.0086</b> |
|                                   | F1 | <b>0.5143</b> | <b>0.6003</b> | <b>0.6863</b> |               |
| SignOrSymptom (gpt-4o-2024-05-13) | P  | 0.4174        | 0.4905        | 0.5637        |               |
|                                   | R  | <b>0.5960</b> | <b>0.7005</b> | <b>0.8050</b> | 0.0089        |
|                                   | F1 | 0.4909        | 0.5770        | 0.6631        |               |

Table 5.29: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Between the two GPT-4o snapshot models, gpt-4o-2024-08-06 achieved the highest utility. Also, gpt-4o-2024-08-06 achieved the lowest test loss across all TRSTR schemes, quite significantly on the TRSTR scheme with 100 EHRs available. As for recall, although it achieved the highest performance for strict and partial scores on the 600 EHRs TRSTR scheme, for the remaining two gpt-4o-2024-05-13 generated SEHRs capable of achieving the highest performance

|                                   |    | Strict        | Partial       | Type          | Loss          |
|-----------------------------------|----|---------------|---------------|---------------|---------------|
| SignOrSymptom (baseline)          | P  | 0.2180        | 0.4087        | <b>0.5995</b> |               |
|                                   | R  | 0.1600        | 0.3000        | 0.4400        | 0.0131        |
|                                   | F1 | 0.1845        | 0.3460        | 0.5075        |               |
| SignOrSymptom (gpt-4o-2024-08-06) | P  | <b>0.3432</b> | <b>0.4274</b> | 0.5116        |               |
|                                   | R  | 0.4870        | 0.6065        | 0.7260        | <b>0.0099</b> |
|                                   | F1 | <b>0.4026</b> | <b>0.5014</b> | <b>0.6002</b> |               |
| SignOrSymptom (gpt-4o-2024-05-13) | P  | 0.2900        | 0.3619        | 0.4339        |               |
|                                   | R  | <b>0.5000</b> | <b>0.6240</b> | <b>0.7480</b> | 0.0112        |
|                                   | F1 | 0.3671        | 0.4581        | 0.5492        |               |

Table 5.30: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Regarding the balance between P and R, across all three TRSTR schemes and metric scores, the same tendency occurs. Between gpt-4o-2024-05-13 and gpt-4o-2024-08-06, the latter is able to find a better balance between precision and recall relatively to the former. The baseline performance shows a better balance, however the performance is worse. Such comparison can be better visualized considering Figure 5.15.

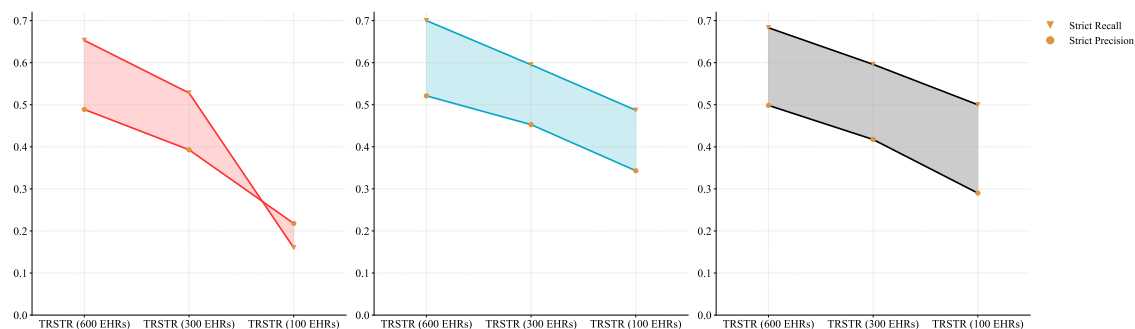


Figure 5.15: Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13. The baseline finds the best balance, but with worse performance.

#### 5.4.3.2 Disease or Syndrome

This section shows the results of BioBERTpt(all) fine-tuned for the Disease or Syndrome STY with 300 SEHRs either generated by gpt-4o-2024-05-13 or gpt-4o-2024-08-06, across the three TRSTR schemes considered for this NER benchmark, with 600, 300, and 100 EHRs. Those results can be seen in Tables 5.31, 5.32, and 5.33, respectively.

|                                       |    | Strict        | Partial       | Type          | Loss          |
|---------------------------------------|----|---------------|---------------|---------------|---------------|
| DiseaseOrSyndrome (baseline)          | P  | 0.5573        | 0.6203        | 0.6834        |               |
|                                       | R  | <b>0.7125</b> | <b>0.7930</b> | <b>0.8736</b> | 0.0041        |
|                                       | F1 | <b>0.6254</b> | <b>0.6961</b> | 0.7669        |               |
| DiseaseOrSyndrome (gpt-4o-2024-08-06) | P  | <b>0.5897</b> | <b>0.6611</b> | <b>0.7326</b> |               |
|                                       | R  | 0.6502        | 0.7289        | 0.8077        | <b>0.0039</b> |
|                                       | F1 | 0.6185        | 0.6934        | <b>0.7683</b> |               |
| DiseaseOrSyndrome (gpt-4o-2024-05-13) | P  | 0.5790        | 0.6432        | 0.7074        |               |
|                                       | R  | 0.6777        | 0.7527        | 0.8278        | 0.0040        |
|                                       | F1 | 0.6245        | 0.6937        | 0.7629        |               |

Table 5.31: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

For this STY, different results were achieved across all TRSTR schemes. Despite being close, both GPT-4o snapshot models were not able to outperform the baseline on the 600 EHR TRSTR scheme, besides gpt-4o-2024-08-06 on all three P metrics and type F1 being able to achieve the lowest loss. On the 300 EHR TRSTR scheme variant, gpt-4o-2024-08-06 achieved the highest performance, mainly for F1 and R metrics. On the 100 EHR TRSTR scheme, gpt-4o-2024-05-13 achieved the best performance, overall.

|                                       |    | Strict        | Partial       | Type          | Loss          |
|---------------------------------------|----|---------------|---------------|---------------|---------------|
| DiseaseOrSyndrome (baseline)          | P  | 0.4937        | 0.5789        | <b>0.6640</b> |               |
|                                       | R  | 0.5733        | 0.6722        | 0.7711        | 0.0055        |
|                                       | F1 | 0.5305        | 0.6220        | <b>0.7136</b> |               |
| DiseaseOrSyndrome (gpt-4o-2024-08-06) | P  | 0.4956        | 0.5661        | 0.6366        |               |
|                                       | R  | <b>0.6245</b> | <b>0.7134</b> | <b>0.8022</b> | <b>0.0044</b> |
|                                       | F1 | <b>0.5527</b> | <b>0.6313</b> | 0.7099        |               |
| DiseaseOrSyndrome (gpt-4o-2024-05-13) | P  | <b>0.5069</b> | <b>0.5794</b> | 0.6518        |               |
|                                       | R  | 0.6026        | 0.6886        | 0.7747        | 0.0046        |
|                                       | F1 | 0.5506        | 0.6293        | 0.7079        |               |

Table 5.32: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Between the two GPT-4o snapshot models, it is difficult to ascertain which achieved the best results. Looking at the last two TRSTR schemes, specifically with 300 and 100 EHRs, the LLM which generated SEHRs with higher utility were gpt-4o-2024-08-06 and gpt-4o-2024-05-13, respectively. That being said, the F1 scores were much closer on the former relatively to the latter. Thus, it can be argued that for these two TRSTR schemes gpt-4o-2024-05-13 performed best. On the TRSTR scheme variant with 600 EHRs, performances were close, however gpt-4o-2024-05-13 achieved the highest strict F1, while on the remaining scores the results were closer. There is no clear better LLM, however it seems that gpt-4o-2024-05-13 performed slightly better than gpt-4o-2024-08-06.

|                                       |    | Strict        | Partial       | Type          | Loss          |
|---------------------------------------|----|---------------|---------------|---------------|---------------|
| DiseaseOrSyndrome (baseline)          | P  | 0.3151        | 0.5034        | 0.6918        |               |
|                                       | R  | 0.0842        | 0.1346        | 0.1850        | 0.0081        |
|                                       | F1 | 0.1329        | 0.2124        | 0.2919        |               |
| DiseaseOrSyndrome (gpt-4o-2024-08-06) | P  | 0.4593        | 0.5425        | 0.6256        |               |
|                                       | R  | 0.4853        | <b>0.5733</b> | <b>0.6612</b> | 0.0055        |
|                                       | F1 | 0.4720        | 0.5574        | 0.6429        |               |
| DiseaseOrSyndrome (gpt-4o-2024-05-13) | P  | <b>0.4944</b> | <b>0.5787</b> | <b>0.6630</b> |               |
|                                       | R  | <b>0.4890</b> | 0.5723        | 0.6557        | <b>0.0054</b> |
|                                       | F1 | <b>0.4917</b> | <b>0.5755</b> | <b>0.6593</b> |               |

Table 5.33: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Regarding the balance between P and R between gpt-4o-2024-05-13 and gpt-4o-2024-08-06, despite the differences being close between the two, the former seems to achieve the best balance of the two. The baseline performance shows a worse balance, especially with 100 EHRs. Such comparison can be better visualized considering Figure 5.16.

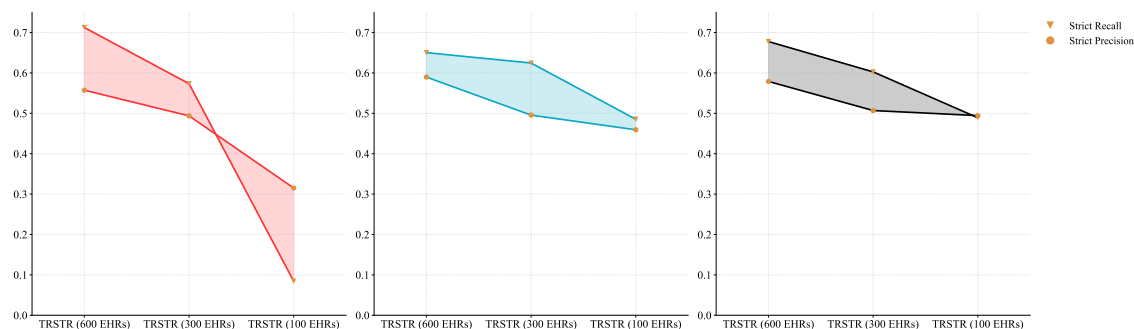


Figure 5.16: Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-05-13 is able to keep strict P and R performance closer than gpt-4o-2024-08-06. The baseline finds the worse balance.

### 5.4.3.3 Laboratory or Test Result

This section shows the results of BioBERTpt(all) fine-tuned for the Laboratory or Test Result STY with 300 SEHRs either generated by gpt-4o-2024-05-13 or gpt-4o-2024-08-06, across the three TRSTR schemes considered for this NER benchmark, with 600, 300, and 100 EHRs. Those results can be seen in Tables 5.34, 5.35, and 5.36, respectively.

|                                     |    | Strict        | Partial       | Type          | Loss          |
|-------------------------------------|----|---------------|---------------|---------------|---------------|
| LabOrTestResult (baseline)          | P  | 0.3627        | 0.4842        | 0.6058        |               |
|                                     | R  | 0.4592        | 0.6131        | <b>0.7671</b> | 0.0071        |
|                                     | F1 | 0.4053        | 0.5411        | 0.6769        |               |
| LabOrTestResult (gpt-4o-2024-08-06) | P  | <b>0.4351</b> | <b>0.5553</b> | <b>0.6755</b> |               |
|                                     | R  | <b>0.4908</b> | <b>0.6265</b> | 0.7621        | <b>0.0069</b> |
|                                     | F1 | <b>0.4613</b> | <b>0.5887</b> | <b>0.7162</b> |               |
| LabOrTestResult (gpt-4o-2024-05-13) | P  | 0.4112        | 0.5272        | 0.6433        |               |
|                                     | R  | 0.4775        | 0.6123        | 0.7471        | 0.0070        |
|                                     | F1 | 0.4419        | 0.5666        | 0.6913        |               |

Table 5.34: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

For this STY, across all TRSTR schemes, both SEHRs generated by gpt-4o-2024-05-13 and gpt-4o-2024-08-06 achieved a positive utility as they, overall, outperformed the baseline across all metrics and scores. As expected, however, such utility increases as the number of real EHRs available decreases. That being said, both SEHR datasets were able to gain a relatively good utility margin on the 600 EHR TRSTR scheme.

|                                     |    | Strict        | Partial       | Type          | Loss          |
|-------------------------------------|----|---------------|---------------|---------------|---------------|
| LabOrTestResult (baseline)          | P  | 0.3266        | 0.4941        | 0.6616        |               |
|                                     | R  | 0.3228        | 0.4884        | 0.6539        | 0.0090        |
|                                     | F1 | 0.3247        | 0.4912        | 0.6577        |               |
| LabOrTestResult (gpt-4o-2024-08-06) | P  | <b>0.3889</b> | <b>0.5128</b> | <b>0.6368</b> |               |
|                                     | R  | 0.4542        | 0.5990        | 0.7438        | <b>0.0074</b> |
|                                     | F1 | <b>0.4190</b> | <b>0.5526</b> | <b>0.6861</b> |               |
| LabOrTestResult (gpt-4o-2024-05-13) | P  | 0.3787        | 0.4926        | 0.6065        |               |
|                                     | R  | <b>0.4676</b> | <b>0.6082</b> | <b>0.7488</b> | 0.0077        |
|                                     | F1 | 0.4185        | 0.5443        | 0.6701        |               |

Table 5.35: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Between the two GPT-4o snapshot models, gpt-4o-2024-08-06 achieved the highest utility. Also, gpt-4o-2024-08-06 achieved the lowest test loss across all TRSTR scheme variants. gpt-4o-2024-08-06 generated SEHRs capable of achieving the highest F1 score and P metrics on all TRSTR schemes, across all scores, except for type P on the 100 EHR TRSTR scheme. gpt-4o-2024-05-13, on the other hand, generated SEHRs which achieved the highest R on the 300 EHR TRSTR scheme.

|                                     |    | Strict        | Partial       | Type          | Loss          |
|-------------------------------------|----|---------------|---------------|---------------|---------------|
| LabOrTestResult (baseline)          | P  | 0.2220        | 0.4260        | <b>0.6300</b> |               |
|                                     | R  | 0.1747        | 0.3353        | 0.4958        | 0.0093        |
|                                     | F1 | 0.1955        | 0.3752        | 0.5549        |               |
| LabOrTestResult (gpt-4o-2024-08-06) | P  | <b>0.3677</b> | <b>0.4858</b> | 0.6038        |               |
|                                     | R  | <b>0.4509</b> | <b>0.5957</b> | <b>0.7404</b> | <b>0.0081</b> |
|                                     | F1 | <b>0.4051</b> | <b>0.5351</b> | <b>0.6652</b> |               |
| LabOrTestResult (gpt-4o-2024-05-13) | P  | 0.3479        | 0.4739        | 0.6000        |               |
|                                     | R  | 0.4110        | 0.5599        | 0.7088        | 0.0083        |
|                                     | F1 | 0.3768        | 0.5133        | 0.6499        |               |

Table 5.36: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Regarding the balance between P and R between gpt-4o-2024-05-13 and gpt-4o-2024-08-06, they are very close. The latter achieves a better balance for the 600 EHRs and 300 EHRs TRSTR scheme variants, while for the 100 EHRs one, gpt-4o-2024-05-13 found the best balance. The baseline performance shows a better balance, however the performance much worse as the number of EHRs available decreases. Such comparison can be better visualized considering Figure 5.17.

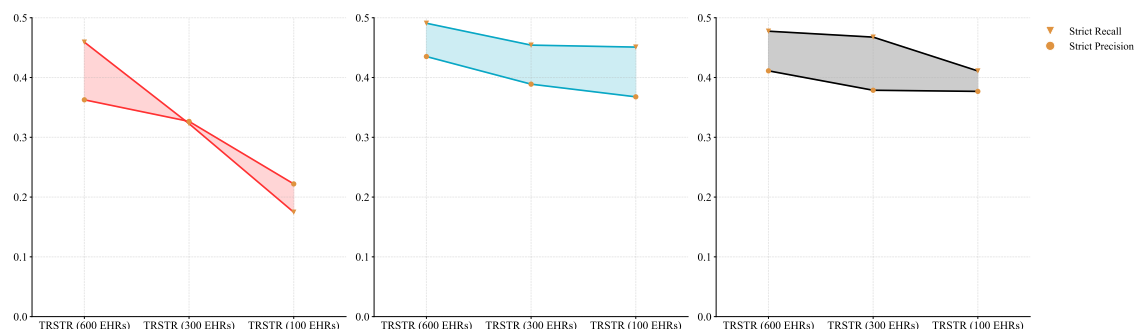


Figure 5.17: Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13 in 2 out of 3 TRSTR schemes. The baseline finds the best balance, but with much worse performance.

#### 5.4.3.4 Quantitative Concept

This section shows the results of BioBERTpt(all) fine-tuned for the Laboratory or Test Result STY with 300 SEHRs either generated by gpt-4o-2024-05-13 or gpt-4o-2024-08-06, across the three TRSTR schemes considered for this NER benchmark, with 600, 300, and 100 EHRs. Those results can be seen in Tables 5.37, 5.38, and 5.39, respectively.

|                                  |    | Strict        | Partial       | Type          | Loss          |
|----------------------------------|----|---------------|---------------|---------------|---------------|
| QuantConcept (baseline)          | P  | <b>0.5778</b> | <b>0.6490</b> | <b>0.7203</b> |               |
|                                  | R  | 0.6957        | 0.7815        | 0.8674        | <b>0.0032</b> |
|                                  | F1 | 0.6313        | <b>0.7091</b> | <b>0.7870</b> |               |
| QuantConcept (gpt-4o-2024-08-06) | P  | 0.5748        | 0.6377        | 0.7006        |               |
|                                  | R  | 0.7191        | 0.7978        | 0.8765        | 0.0033        |
|                                  | F1 | <b>0.6389</b> | 0.7088        | 0.7787        |               |
| QuantConcept (gpt-4o-2024-05-13) | P  | 0.5590        | 0.6132        | 0.6673        |               |
|                                  | R  | <b>0.7451</b> | <b>0.8173</b> | <b>0.8895</b> | 0.0034        |
|                                  | F1 | 0.6388        | 0.7007        | 0.7625        |               |

Table 5.37: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Regarding this STY, for the TRSTR schemes with 600 and 300 EHRs, the baseline achieves the best performance on the P and R metrics, except for the highest strict F1 on the 600 EHR TRSTR scheme, which was achieved by the SEHRs generated by gpt-4o-2024-08-06. For these TRSTR schemes, the baseline also achieved the lowest test loss. That being said, the performances between the baseline and the two GPT-4o snapshot models were very close, except on the 100 EHR TRSTR scheme. gpt-4o-2024-08-06 outperformed gpt-4o-2024-05-13 and the baseline on such scheme.

|                                  |    | Strict        | Partial       | Type          | Loss          |
|----------------------------------|----|---------------|---------------|---------------|---------------|
| QuantConcept (baseline)          | P  | <b>0.5532</b> | <b>0.6334</b> | <b>0.7135</b> |               |
|                                  | R  | 0.6554        | 0.7503        | 0.8453        | <b>0.0035</b> |
|                                  | F1 | <b>0.6000</b> | <b>0.6869</b> | <b>0.7738</b> |               |
| QuantConcept (gpt-4o-2024-08-06) | P  | 0.5415        | 0.6115        | 0.6814        |               |
|                                  | R  | 0.6697        | 0.7562        | 0.8427        | 0.0036        |
|                                  | F1 | 0.5988        | 0.6762        | 0.7535        |               |
| QuantConcept (gpt-4o-2024-05-13) | P  | 0.5023        | 0.5669        | 0.6314        |               |
|                                  | R  | <b>0.6983</b> | <b>0.7880</b> | <b>0.8778</b> | 0.0040        |
|                                  | F1 | 0.5843        | 0.6594        | 0.7345        |               |

Table 5.38: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Between the two GPT-4o snapshot models, gpt-4o-2024-08-06 achieved the highest utility. Also, gpt-4o-2024-08-06 achieved the lowest test loss across all TRSTR schemes, except on the 100 EHR TRSTR scheme variant where their loss is approximately the same. gpt-4o-2024-08-06 generated SEHRs capable of achieving the highest F1 score and P metrics on all TRSTR schemes, across all scores, relative to gpt-4o-2024-05-13. gpt-4o-2024-05-13, on the other hand, generated SEHRs which achieved the highest R except on the 100 EHR TRSTR scheme variant.

|                                  |    | Strict        | Partial       | Type          | Loss          |
|----------------------------------|----|---------------|---------------|---------------|---------------|
| QuantConcept (baseline)          | P  | 0.3033        | 0.5082        | <b>0.7131</b> |               |
|                                  | R  | 0.0962        | 0.1612        | 0.2263        | 0.0060        |
|                                  | F1 | 0.1461        | 0.2448        | 0.3435        |               |
| QuantConcept (gpt-4o-2024-08-06) | P  | <b>0.4990</b> | <b>0.5784</b> | 0.6578        |               |
|                                  | R  | <b>0.6333</b> | <b>0.7341</b> | <b>0.8349</b> | <b>0.0045</b> |
|                                  | F1 | <b>0.5582</b> | <b>0.6470</b> | <b>0.7358</b> |               |
| QuantConcept (gpt-4o-2024-05-13) | P  | 0.4468        | 0.5234        | 0.6000        |               |
|                                  | R  | 0.5956        | 0.6977        | 0.7997        | <b>0.0045</b> |
|                                  | F1 | 0.5106        | 0.5981        | 0.6856        |               |

Table 5.39: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Regarding the balance between P and R between gpt-4o-2024-05-13 and gpt-4o-2024-08-06, the latter achieved a better balance between the aforementioned strict metrics. The baseline performance shows a relatively good balance. The baseline’s performance became much worse as the number of EHRs available decreased to 100, in contrast with gpt-4o-2024-05-13 and gpt-4o-2024-08-06. Such comparison can be better visualized considering Figure 5.18.

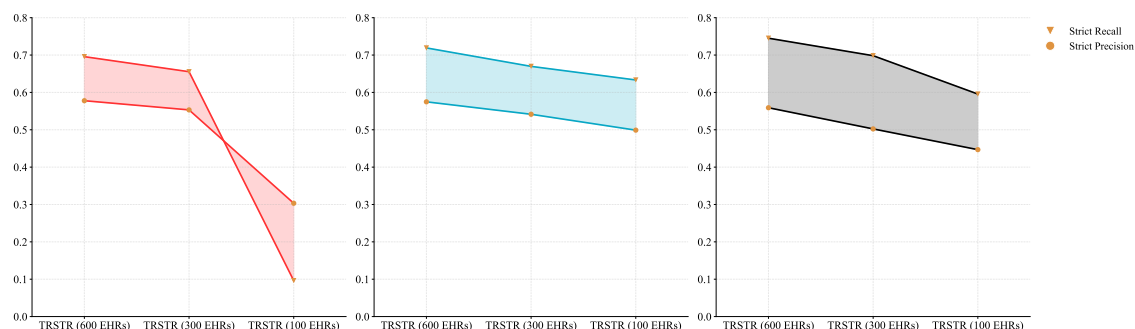


Figure 5.18: Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13. The baseline finds a good balance, but with a much worse performance on the 100 EHR TRSTR scheme.

#### 5.4.3.5 Diagnostic Procedure

This section shows the results of BioBERTpt(all) fine-tuned for the Diagnostic Procedure STY with 300 SEHRs either generated by gpt-4o-2024-05-13 or gpt-4o-2024-08-06, across the three TRSTR schemes considered for this NER benchmark, with 600, 300, and 100 EHRs. Those results can be seen in Tables 5.40, 5.41, and 5.42, respectively.

|                                    |    | Strict        | Partial       | Type          | Loss          |
|------------------------------------|----|---------------|---------------|---------------|---------------|
| DiagnosticProc (baseline)          | P  | 0.5959        | 0.6454        | 0.6948        |               |
|                                    | R  | 0.7316        | <b>0.7924</b> | <b>0.8532</b> | <b>0.0020</b> |
|                                    | F1 | 0.6568        | 0.7114        | 0.7659        |               |
| DiagnosticProc (gpt-4o-2024-08-06) | P  | <b>0.6397</b> | <b>0.6790</b> | <b>0.7183</b> |               |
|                                    | R  | <b>0.7418</b> | 0.7873        | 0.8329        | <b>0.0020</b> |
|                                    | F1 | <b>0.6870</b> | <b>0.7292</b> | <b>0.7714</b> |               |
| DiagnosticProc (gpt-4o-2024-05-13) | P  | 0.5598        | 0.6014        | 0.6429        |               |
|                                    | R  | 0.7342        | 0.7886        | 0.8430        | 0.0023        |
|                                    | F1 | 0.6353        | 0.6824        | 0.7295        |               |

Table 5.40: BioBERTpt(all) results fine-tuned with an augmented training corpus of 600 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

For this STY, across all TRSTR schemes, the SEHRs generated by gpt-4o-2024-08-06 achieved a positive utility as, overall, outperformed the baseline for the majority of all metrics and scores. gpt-4o-2024-05-13 underperformed as the number of EHRs available for training increased.

|                                    |    | Strict        | Partial       | Type          | Loss          |
|------------------------------------|----|---------------|---------------|---------------|---------------|
| DiagnosticProc (baseline)          | P  | 0.5135        | 0.5987        | <b>0.6839</b> |               |
|                                    | R  | 0.5797        | 0.6759        | 0.7722        | 0.0029        |
|                                    | F1 | 0.5446        | 0.6350        | <b>0.7253</b> |               |
| DiagnosticProc (gpt-4o-2024-08-06) | P  | <b>0.5601</b> | <b>0.6202</b> | 0.6803        |               |
|                                    | R  | 0.6253        | 0.6924        | 0.7595        | <b>0.0023</b> |
|                                    | F1 | <b>0.5909</b> | <b>0.6543</b> | 0.7177        |               |
| DiagnosticProc (gpt-4o-2024-05-13) | P  | 0.5192        | 0.5687        | 0.6182        |               |
|                                    | R  | <b>0.6506</b> | <b>0.7127</b> | <b>0.7747</b> | 0.0025        |
|                                    | F1 | 0.5775        | 0.6326        | 0.6876        |               |

Table 5.41: BioBERTpt(all) results fine-tuned with an augmented training corpus of 300 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Between the two GPT-4o snapshot models, gpt-4o-2024-08-06 achieved the highest utility, as well as the lowest test loss across all TRSTR schemes. gpt-4o-2024-08-06 generated SEHRs capable of achieving the highest performance on the majority of the F1 and P metrics on all TRSTR schemes, across all scores. gpt-4o-2024-05-13 generated SEHRs which achieved the highest R on the 300 EHR TRSTR scheme variant. The R scores of gpt-4o-2024-08-06 did not fall too short.

|                                    |    | Strict        | Partial       | Type          | Loss          |
|------------------------------------|----|---------------|---------------|---------------|---------------|
| DiagnosticProc (baseline)          | P  | 0.0000        | 0.2500        | 0.5000        |               |
|                                    | R  | 0.0000        | 0.0025        | 0.0051        | 0.0060        |
|                                    | F1 | 0.0000        | 0.0050        | 0.0100        |               |
| DiagnosticProc (gpt-4o-2024-08-06) | P  | <b>0.4160</b> | <b>0.4737</b> | <b>0.5315</b> |               |
|                                    | R  | 0.5013        | <b>0.5709</b> | <b>0.6405</b> | <b>0.0035</b> |
|                                    | F1 | <b>0.4546</b> | <b>0.5178</b> | <b>0.5809</b> |               |
| DiagnosticProc (gpt-4o-2024-05-13) | P  | 0.3810        | 0.4248        | 0.4686        |               |
|                                    | R  | <b>0.5063</b> | 0.5646        | 0.6228        | 0.0038        |
|                                    | F1 | 0.4348        | 0.4848        | 0.5348        |               |

Table 5.42: BioBERTpt(all) results fine-tuned with an augmented training corpus of 100 real EHRs and 300 SEHRs generated either using gpt-4o-2024-05-13 or gpt-4o-2024-08-06.

Regarding the balance between P and R between gpt-4o-2024-05-13 and gpt-4o-2024-08-06,

the latter achieves a better balance across all TRSTR schemes, while gpt-4o-2024-05-13 achieved the worse balance. The baseline performance shows a slight worse balance relative to gpt-4o-2024-08-06 if we exclude the performance on the 100 EHR TRSTR scheme variant. Such comparison can be better visualized considering Figure 5.19.

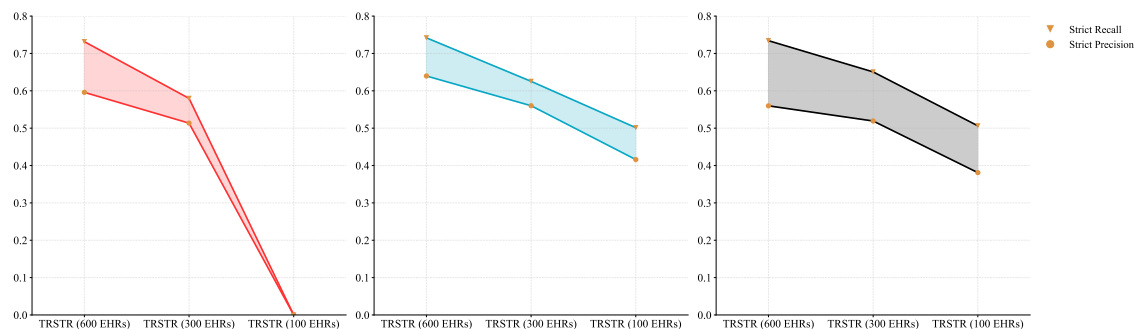


Figure 5.19: Parallel coordinates plot for the differences between strict P and R for the baseline (left), gpt-4o-2024-08-06 (center), and gpt-4o-2024-05-13 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than gpt-4o-2024-05-13 in 2 out of 3 TRSTR schemes. The baseline finds the best balance, but with much worse performance.

## 5.5 SDG final pipeline with different PLM encoder applied to PortugueseClinicalNER

The final SDG pipeline, developed in Section 5.4, showed being capable of generating SEHRs and NER annotations which were able to increase performance on SemClinBr, a benchmark which had clinical texts quite complex and with expert-level annotations. Despite such achievement giving ground work to answer the research question, we went a step further and applied the SDG pipeline to a different NER benchmark, unbiased of all the work developed to increase utility on SemClinBr. Not only that, but we also replaced BioBERTpt(all) by an unbiased PLM. MediAlbertina PT-PT with 900M parameters was chosen.

Regarding the following sections, Section 5.5.1 will dive into the results achieved in terms of privacy disclosure, resemblance, and utility. The final prompt achieved in Section 5.4 was properly adjusted, while keeping everything else the same, from the GPT-4o snapshot model and its hyperparameter choice. Section 5.5.2 shows the results achieved for NER.

### 5.5.1 SEHRs document and annotation generation

Considering the usage of the SDG pipeline developed in Section 5.4, the main change which is required to do is the prompt. More specifically, the Annotation Tag Definitions and the file format in the user instruction. The prompt can be seen below. The results which drove the STY selection are shown in Section 5.5.2

Besides that, however, there is another important component, the set of CONCEPTS given to each prompt randomly. Further details can be seen in Section 4.2.3.3. Below it can be seen the modified version Prompt GPT-4o adjusted to this new NER dataset.

**[System instruction]** ""You are an advanced AI language model that generates and annotates synthetic health reports in Portuguese. Your task is to generate high-quality health reports and provide accurate Named Entity Recognition annotations based on the provided definitions and examples.

### Guidelines:

1. Language: Generate text in European Portuguese.
2. Health Report Context: Ensure the text reflects typical health report scenarios. Follow the structure of the reports uploaded to the system instructions.
3. Annotations: Annotate entities accurately using the provided definitions and examples. Follow the format used in the file examples uploaded to the system instructions.

### Annotation Tag Definitions:

1. THER: This tag identifies references to therapeutic interventions, including medications, treatments, and procedures used for patient management.
2. C: This tag highlights descriptions of medical conditions, symptoms, or diagnoses.
3. OBS: This tag is used for supplementary clinical observations that do not necessarily describe a medical condition but provide additional information about the patient's state.
4. DT: This tag marks temporal information, including specific dates, timeframes, or durations relevant to the patient's history or condition.
5. CH: This tag captures descriptive details that characterize conditions, symptoms, or findings.
6. R: This tag refers to the outcomes or findings from diagnostic tests, examinations, or treatments.

**[User instructions]** ""Considering the system instructions given and the files uploaded to it, generate a complete, extensive, and high-quality electronic health report, with around 40 entities for tag annotation across all annotation tags, for a patient with the following condition: "CONCEPT". Generate annotations according to the annotation tag definitions, while considering all the tags mentioned in it. Also, pay close attention to how the tags were used in the uploaded file examples. Use several abbreviations just like in the file examples uploaded. Generate a CoNLL-2003 file format like the text files I uploaded. Your output/response should only have the generated file.""

**Prompt:** GPT-4o prompt adjusted adapted from the SemClinBr version by changing the Annotation Tag Definitions, Language, and the file format in which the output should come.

Considering the size of the training corpus, with 202 EHRs, and for budget reasons, the SDG pipeline was used to generate a final SEHR dataset with 100 SEHRs. Such 100 SEHRs resulted in a total of 1,396 sentences. As for the number of annotations per STY, such results can be seen in Table 5.43.

| Semantic Type                 | gpt-4o-2024-08-06 |
|-------------------------------|-------------------|
| Additional Observations (OBS) | 312               |
| Condition (C)                 | 1,143             |
| Result (R)                    | 440               |
| Therapeutics (THER)           | 570               |
| DateTime (DT)                 | 561               |
| Characteristics (CH)          | 183               |

Table 5.43: Number of occurrences of each STY for the SEHR dataset generated using gpt-4o-2024-08-06

As for privacy disclosure quantitative measurements, because there is no other SEHR dataset to compare with, instead of using a dumbbell plot, a simple table was chosen to show the results. Such BLEU scores, from bigrams up to 5-grams can be seen in Table 5.44. Such results were achieved by comparing the 100 SEHRs with their respective references, which in this case are either training, validation, or test EHRs.

| BLEU score measures | Training refs | Dev refs | Test refs |
|---------------------|---------------|----------|-----------|
| BLEU-2              | 0.6162        | 0.4554   | 0.4610    |
| BLEU-3              | 0.4100        | 0.2560   | 0.2644    |
| BLEU-4              | 0.2531        | 0.1276   | 0.1426    |
| BLEU-5              | 0.1440        | 0.0607   | 0.0665    |

Table 5.44: BLEU score measures obtained considering as references the training, development, and test EHRs

As with the SemClinBr BLEU scores, such scores are higher when considered as references the training EHRs when compared with the BLEU scores which consider as references the validation and test EHRs. This shows the impact of the usage of training files to augment the SDG pipeline, as well as the temperature and top-p threshold value choice.

As for resemblance, it was interesting to see how the synthetic sentence embeddings compared to the real sentence embeddings in a different benchmark. The mapping using PCA and t-SNE can be seen in Figure 5.20.

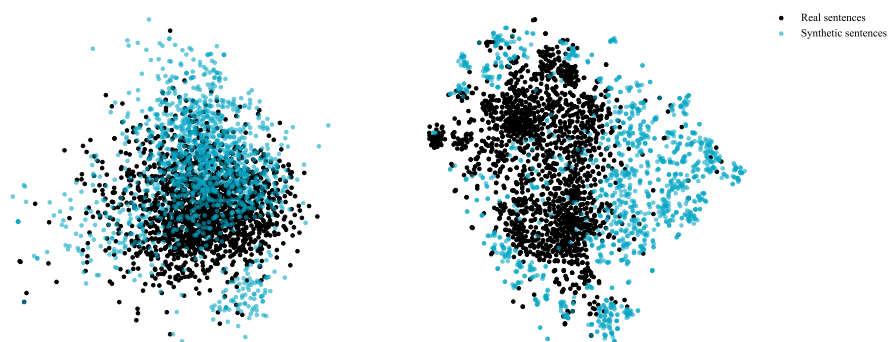


Figure 5.20: Scatterplots showing the sentence embedding distribution of the real and synthetic sentences in a 2-dimensional latent space computed using PCA (left) and t-SNE (right). KL divergence for PCA is equal to 0.21958 and for t-SNE the KL divergence is equal to 11.24558.

These results are, indeed, interesting, but surprising as well. With PCA the sentence embeddings seem to overlap more than in t-SNE, however, the latter shows distinct groups of sentence embeddings. Also, it is interesting to see that it appears there is a small overlap when computing latent variables using t-SNE. Comparing the t-SNE results with SemClinBr, more specifically with the results of the same GPT-4o snapshot model in Section 5.4.2, it seems to exist a bigger overlap between sentence embeddings in SemClinBr relative to the sentence embeddings of Portuguese-ClinicalNER. As for PCA, while with SemClinBr there were two well-defined groups of sentence embeddings, with PortugueseClinicalNER there is no such pattern.

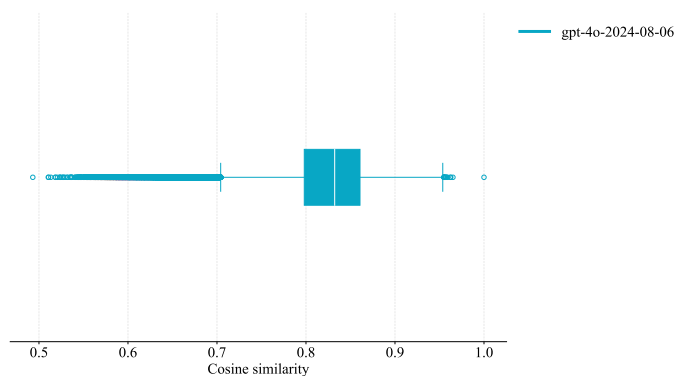


Figure 5.21: Boxplot of the cosine similarity scores between real and synthetic sentences generated by gpt-4o-2024-08-06. The average cosine similarity is equal to 0.8252.

Analyzing these results, it was intriguing to check how the cosine similarity evolved. Meaning, if it follows the apparent lack of overlapping shown with the mapping made with t-SNE. To check this, Figure 5.21 shows a boxplot of all cosine similarities.

Despite the boxplots for SemClinBr showing the averages, in Figure 5.11, Figure 5.21 shows cosine similarities much higher, resulting in an average cosine similarity of 0.8252. So, despite

the apparent lack of overlapping of sentence embeddings using t-SNE showed in Figure 5.20, they seem to have a higher similarity than those generated to augment the training corpus of SemClinBr.

Considering the KL divergences, the ones for PortugueseClinicalNER, in Figure 5.20, were higher than those in Figures 5.9 and 5.10. Despite their entropy being higher with PortugueseClinicalNER, the cosine similarities are much higher relative to those in SemClinBr. Thus, we can see that KL divergence and cosine similarity give different conclusions. In other words, despite the entropy between the sentence embedding's latent variables being higher (more distinct distributions of real and synthetic sentences, thus worse KL divergence) in PortugueseClinicalNER, the cosine similarity was also higher (higher similarity between real and synthetic sentence embeddings, thus better).

### 5.5.2 Participant extraction

To adjust the system instruction component of the prompt, more specifically the Annotation Tag Definitions, it was necessary to choose which STYs from the PortugueseClinicalNER benchmark should the SDG pipeline focus on. More details can be read in Section 4.3.2.1. The results used to choose the STYs are shown below.

Thus, the first results achieved on this new benchmark were not related to the SDG pipeline, but rather to extracting participants for all STYs available in PortugueseClinicalNER. Some hyperparameter tuning was made to achieve better performance, overall, as well as reducing as much as possible overfitting without compromising performance on the development set. Learning curves are not shown because only two epochs were considered. For all STYs, overall, after hyperparameter tuning it still existed some overfitting. The first epoch showed lower validation loss relative to the training loss. On the second epoch the opposite occurred.

After tuning the model's hyperparameters, `medialbertina_pt-pt_900m` was fine-tuned for all STYs, one at a time. The results can be seen in Table 5.45.

|                                          |    | Strict        | Partial       | Type          | Loss          |
|------------------------------------------|----|---------------|---------------|---------------|---------------|
| Characterization - CH (baseline)         | P  | <b>0.3538</b> | <b>0.4359</b> | <b>0.5179</b> |               |
|                                          | R  | <b>0.4367</b> | <b>0.5380</b> | <b>0.6392</b> | <b>0.0263</b> |
|                                          | F1 | <b>0.3909</b> | <b>0.4816</b> | <b>0.5722</b> |               |
| Test - T (baseline)                      | P  | 0.7267        | 0.7898        | 0.8529        |               |
|                                          | R  | 0.8231        | 0.8946        | 0.9660        | 0.0163        |
|                                          | F1 | 0.7719        | 0.8389        | 0.9059        |               |
| Evolution - EV (baseline)                | P  | 0.7969        | 0.8320        | 0.8672        |               |
|                                          | R  | 0.8947        | 0.9342        | 0.9737        | 0.0050        |
|                                          | F1 | 0.8430        | 0.8802        | 0.9174        |               |
| Genetics - G (baseline)                  | P  | 0.6667        | 0.7333        | 0.8000        |               |
|                                          | R  | 0.8333        | 0.9167        | 1.0000        | 0.0011        |
|                                          | F1 | 0.7407        | 0.8148        | 0.8889        |               |
| Anatomical Site - AS (baseline)          | P  | 0.7866        | 0.8329        | 0.8792        |               |
|                                          | R  | 0.8407        | 0.8901        | 0.9396        | 0.0170        |
|                                          | F1 | 0.8127        | 0.8606        | 0.9084        |               |
| Negation - N (baseline)                  | P  | 0.9796        | 0.9847        | 0.9898        |               |
|                                          | R  | 0.9697        | 0.9747        | 0.9798        | 0.0005        |
|                                          | F1 | 0.9746        | 0.9797        | 0.9848        |               |
| Additional Observations - OBS (baseline) | P  | <b>0.3636</b> | <b>0.5455</b> | <b>0.7273</b> |               |
|                                          | R  | <b>0.2105</b> | <b>0.3158</b> | <b>0.4211</b> | <b>0.0123</b> |
|                                          | F1 | <b>0.2667</b> | <b>0.4000</b> | <b>0.5333</b> |               |
| Condition - C (baseline)                 | P  | <b>0.6811</b> | <b>0.7566</b> | <b>0.8322</b> |               |
|                                          | R  | <b>0.7664</b> | <b>0.8514</b> | <b>0.9364</b> | <b>0.0482</b> |
|                                          | F1 | <b>0.7212</b> | <b>0.8012</b> | <b>0.8813</b> |               |
| Results - R (baseline)                   | P  | <b>0.4379</b> | <b>0.5503</b> | <b>0.6627</b> |               |
|                                          | R  | <b>0.5714</b> | <b>0.7181</b> | <b>0.8649</b> | <b>0.0383</b> |
|                                          | F1 | <b>0.4958</b> | <b>0.6231</b> | <b>0.7504</b> |               |
| DateTime - DT (baseline)                 | P  | <b>0.6627</b> | <b>0.7610</b> | <b>0.8594</b> |               |
|                                          | R  | <b>0.7269</b> | <b>0.8348</b> | <b>0.9427</b> | <b>0.0151</b> |
|                                          | F1 | <b>0.6933</b> | <b>0.7962</b> | <b>0.8992</b> |               |
| Therapeutics - THER (baseline)           | P  | <b>0.5896</b> | <b>0.6828</b> | <b>0.7761</b> |               |
|                                          | R  | <b>0.7453</b> | <b>0.8632</b> | <b>0.9811</b> | <b>0.0050</b> |
|                                          | F1 | <b>0.6583</b> | <b>0.7625</b> | <b>0.8667</b> |               |
| Value - V (baseline)                     | P  | 0.7826        | 0.8406        | 0.8986        |               |
|                                          | R  | 0.8308        | 0.8923        | 0.9538        | 0.0028        |
|                                          | F1 | 0.8060        | 0.8657        | 0.9254        |               |
| Route of Administration - RA (baseline)  | P  | 0.7778        | 0.8333        | 0.8889        |               |
|                                          | R  | 0.7000        | 0.7500        | 0.8000        | 0.0003        |
|                                          | F1 | 0.7368        | 0.7895        | 0.8421        |               |

Table 5.45: Baseline results for all STYs of medialbertina\_pt-pt\_900m fine-tuned with a training corpus of 202 EHRs. In bold are the STYs and their respective baseline performances chosen to be augmented

STYs with the lowest strict F1 scores were selected to be augmented using the SDG pipeline developed for SemClinBr. Considering the baseline performances shown in Table 5.45, the choice for STYs was the following: Characterization (CH), Additional Observations (OBS), Condition (C), Results (R), DateTime (DT), and Therapeutics (THER).

With such selection made, the training corpus was augmented with 100 SEHRs generated, achieving the results that were detailed in Section 5.5.1. As for the performance of medialbertina\_pt-pt\_900m fine-tuned with the augmented dataset developed, Figure 5.22 shows a radial plot of the strict F1 performances across the STYs selected.

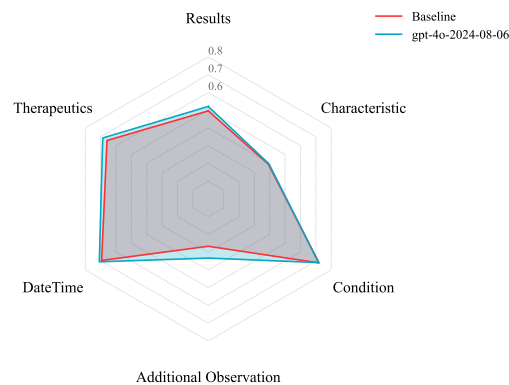


Figure 5.22: Radial plot of strict F1 scores across all 6 STYs selected relative to the SEHR dataset generated using as LLM gpt-4o-2024-08-06, and the respective baselines. Utility changes depending on which STY is being analyzed. However, overall results show a positive utility with the SEHRs enabling a strict F1 increase across 5 out of 6 STYs. Highest utility achieved for Additional Observations.

Looking at the strict F1 performance across all STYs, it is noticeable a performance increase, with emphasis to Results, THER, and OBS. Condition was the only STY which did not see an improvement relative to the baseline performance. Out of all STYs, OBS was the one which saw the highest utility margin.

Similarly to what was done in Section 5.4.3, it is interesting to check how the SEHR dataset generated performed regarding the balance between strict P and R. For that purpose, Figure 5.23 comprises two parallel coordinates plot for both baseline and augmented performance considering the aforementioned metrics, using the shaded area as a perception cue to better and easily understand which performance shows the best balance.

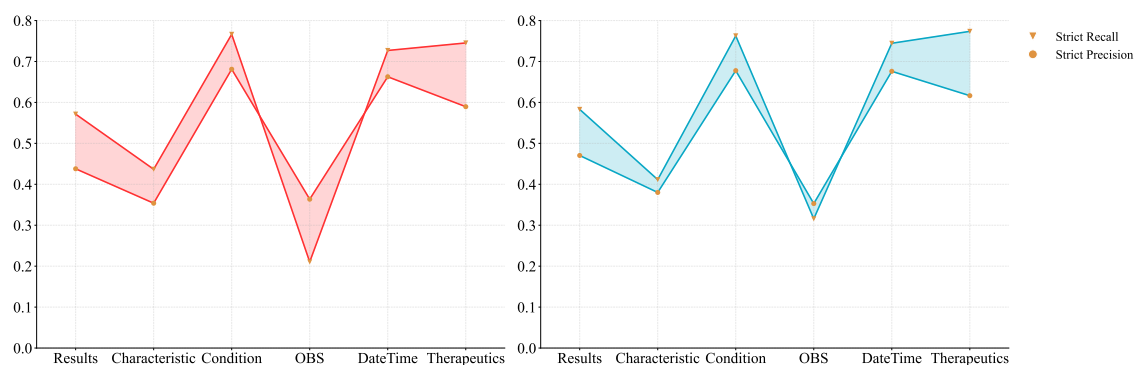


Figure 5.23: Parallel coordinates plot for the differences between strict P and R for the baseline (left) and gpt-4o-2024-08-06 (right). gpt-4o-2024-08-06 is able to keep strict P and R performance closer than the baseline.

Looking at Figure 5.23, although the differences depend on the STY, overall it can be perceived a better balance between strict P and R for medialbertina\_pt-pt\_900m fine-tuned with the

augmented training corpus relative to the baseline performance.

Tables 5.46, 5.47, 5.48, 5.49, 5.50, and 5.51 show the final results achieved for the Portuguese-ClinicalNER benchmark, the test made to the SDG pipeline, specifically for the Results, Characterization, Condition, Additional Observations, DateTime, and Therapeutics, respectively.

|                             |    | Strict        | Partial       | Type          | Loss          |
|-----------------------------|----|---------------|---------------|---------------|---------------|
| Results (baseline)          | P  | 0.4379        | 0.5503        | 0.6627        |               |
|                             | R  | 0.5714        | <b>0.7181</b> | <b>0.8649</b> | 0.0383        |
|                             | F1 | 0.4958        | 0.6231        | 0.7504        |               |
| Results (gpt-4o-2024-08-06) | P  | <b>0.4704</b> | <b>0.5794</b> | <b>0.6885</b> |               |
|                             | R  | <b>0.5830</b> | <b>0.7181</b> | 0.8533        | <b>0.0375</b> |
|                             | F1 | <b>0.5207</b> | <b>0.6414</b> | <b>0.7621</b> |               |

Table 5.46: medialbertina\_pt-pt\_900m results fine-tuned for the Results STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed.

|                                      |    | Strict        | Partial       | Type          | Loss          |
|--------------------------------------|----|---------------|---------------|---------------|---------------|
| Characterization (baseline)          | P  | 0.3538        | 0.4359        | 0.5179        |               |
|                                      | R  | <b>0.4367</b> | <b>0.5380</b> | <b>0.6392</b> | 0.0263        |
|                                      | F1 | 0.3909        | 0.4816        | 0.5722        |               |
| Characterization (gpt-4o-2024-08-06) | P  | <b>0.3801</b> | <b>0.4854</b> | <b>0.5906</b> |               |
|                                      | R  | 0.4114        | 0.5253        | <b>0.6392</b> | <b>0.0256</b> |
|                                      | F1 | <b>0.3951</b> | <b>0.5046</b> | <b>0.6140</b> |               |

Table 5.47: medialbertina\_pt-pt\_900m results fine-tuned for the Characterization STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed.

|                               |    | Strict        | Partial       | Type          | Loss          |
|-------------------------------|----|---------------|---------------|---------------|---------------|
| Condition (baseline)          | P  | <b>0.6811</b> | <b>0.7566</b> | <b>0.8322</b> |               |
|                               | R  | <b>0.7664</b> | <b>0.8514</b> | <b>0.9364</b> | 0.0482        |
|                               | F1 | <b>0.7212</b> | <b>0.8012</b> | <b>0.8813</b> |               |
| Condition (gpt-4o-2024-08-06) | P  | 0.6777        | 0.7500        | 0.8223        |               |
|                               | R  | 0.7626        | 0.8439        | 0.9252        | <b>0.0463</b> |
|                               | F1 | 0.7177        | 0.7942        | 0.8707        |               |

Table 5.48: medialbertina\_pt-pt\_900m results fine-tuned for the Condition STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed.

|                                             |    | Strict        | Partial       | Type          | Loss          |
|---------------------------------------------|----|---------------|---------------|---------------|---------------|
| Additional Observations (baseline)          | P  | <b>0.3636</b> | <b>0.5455</b> | <b>0.7273</b> |               |
|                                             | R  | 0.2105        | 0.3158        | 0.4211        | 0.0123        |
|                                             | F1 | 0.2667        | 0.4000        | <b>0.5333</b> |               |
| Additional Observations (gpt-4o-2024-08-06) | P  | 0.3529        | 0.4559        | 0.5588        |               |
|                                             | R  | <b>0.3158</b> | <b>0.4079</b> | <b>0.5000</b> | <b>0.0117</b> |
|                                             | F1 | <b>0.3333</b> | <b>0.4306</b> | 0.5278        |               |

Table 5.49: medialbertina\_pt-pt\_900m results fine-tuned for the OBS STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed.

|                              |    | Strict        | Partial       | Type          | Loss          |
|------------------------------|----|---------------|---------------|---------------|---------------|
| DateTime (baseline)          | P  | 0.6627        | 0.7610        | 0.8594        |               |
|                              | R  | 0.7269        | 0.8348        | 0.9427        | 0.0151        |
|                              | F1 | 0.6933        | 0.7962        | 0.8992        |               |
| DateTime (gpt-4o-2024-08-06) | P  | <b>0.6760</b> | <b>0.7740</b> | <b>0.8720</b> |               |
|                              | R  | <b>0.7445</b> | <b>0.8524</b> | <b>0.9604</b> | <b>0.0139</b> |
|                              | F1 | <b>0.7086</b> | <b>0.8113</b> | <b>0.9140</b> |               |

Table 5.50: medialbertina\_pt-pt\_900m results fine-tuned for the DateTime STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed.

|                                  |    | Strict        | Partial       | Type          | Loss          |
|----------------------------------|----|---------------|---------------|---------------|---------------|
| Therapeutics (baseline)          | P  | 0.5896        | 0.6828        | <b>0.7761</b> |               |
|                                  | R  | 0.7453        | 0.8632        | <b>0.9811</b> | <b>0.0050</b> |
|                                  | F1 | 0.6583        | 0.7625        | <b>0.8667</b> |               |
| Therapeutics (gpt-4o-2024-08-06) | P  | <b>0.6165</b> | <b>0.6917</b> | 0.7669        |               |
|                                  | R  | <b>0.7736</b> | <b>0.8679</b> | 0.9623        | 0.0076        |
|                                  | F1 | <b>0.6862</b> | <b>0.7699</b> | 0.8536        |               |

Table 5.51: medialbertina\_pt-pt\_900m results fine-tuned for the THER STY with an augmented training corpus of 202 real EHRs and 100 SEHRs generated using gpt-4o-2024-08-06 as part of the SDG pipeline developed.

## 5.6 Summary

We developed a SDG pipeline composed by a system/user instruction prompt augmented with File Search, using GPT-4o. With it, we were capable of generating and annotating SEHRs with lower privacy disclosure risks. Not only that, but with the SEHRs given by it, we were also able to achieve higher NER performance across the several data shortage scenarios, the two NER datasets, and two PLM encoders used.

A comparative study was conducted to compare UMLS and GPT-4o annotations for the same sample of 35 pSEHRs. Across the vast majority of STYs and TRSTR scheme variants, GPT-4o was the better annotator. Those gains being more evident as data shortage scenarios had less and less available real EHRs for training. We went forward with GPT-4o as an generator and annotator of SEHRs.

Using CONCEPT sampling (Prompt GPT-4o adjusted), we achieved the highest strict F1 performance, until this point. With it, we were also able to reduce the risks of privacy disclosures and generate more varied SEHRs. This can be said given the lower BLEU scores achieved in Figures 5.4. Final results for SemClinBr showed that NER performance decreases as data shortage becomes less of a problem, while simultaneously outperforming the baselines. But most importantly, we were able to dramatically close the NER performance gap between the 100 EHRs and 300 EHRs data shortage scenarios by augmenting the former with 300 SEHRs. Those insights can be seen in Figure 5.14.

PortugueseClinicalNER and MediAlbertina PT-PT with 900M parameters were chosen to test the SDG pipeline as an unbiased NER dataset and unbiased PLM. NER results showed a positive utility with performance gains, especially for the OBS STY, which had a strict F1 boost from 26.7% to 33.3%, relative to the baseline. This showed that the SDG pipeline developed in this work has the potential to help achieve higher NER performance for different use cases from the one it was used to develop it.

## Chapter 6

# Conclusions and Future Work

This section aims to discuss all the results achieved with the work done for this dissertation and reflect on how the work can lay the foundation for further work. Such details are present in Sections 6.2 and 6.3, respectively. Section 6.1 gives a brief recap on the overall line of work, as well as on the research question.

### 6.1 Summary of the work developed

Regarding the research question detailed in 1.2, considering the final results achieved, we can answer it positively. In other words, with the final SDG pipeline developed, we were able to increase NER performance in the medical domain. For that, we developed a written prompt with both system and user instructions. On top of that, we utilized File Search, a tool somewhat similar to a RAG system, to augment even more the written prompt with context from EHR examples. The CONCEPT sampling component and the EHR examples were also very important for the higher NER performance achieved. Lower privacy disclosure risks were also achieved using CONCEPT sampling.

This work started with developing several initial prompts while comparing and assessing their outputs for a specific disease with an oncology expert. This led to Prompt 5, which was used with gpt-3.5-turbo to generate 1,200 pSEHRs and bSEHRs. A UMLS-based tool was developed to weakly annotate those SEHRs.

Considering the NER performances, a hypothesis was made that the annotating process was impacting the utility more than the generation process. A comparative study with the same pSEHRs annotated with UMLS and GPT-4o was made. We also generated and annotated SEHRs with GPT-4o to check if, for the same CONCEPT sampling of conditions, GPT-4o generated better SEHRs. The setup with GPT-4o as generator and annotator was chosen.

We decided to make further developments to the prompt and check the improved prompt's results in a TRSTR variant with 100 EHRs. We either used CONCEPT sampling (Prompt GPT-4o

adjusted) or not (Prompt GPT-4o adjusted v2). Prompt GPT-4o adjusted showed better utility with lower privacy disclosure risks, showing such balance is possible. Using CONCEPT sampling, we fine-tuned the GPT-4o hyperparameters, temperature and top-p threshold. We generated 300 SEHRs using each of the two GPT-4o snapshot models, gpt-4o-2024-05-13 and gpt-4o-2024-08-06. Thus, the final SDG pipeline consists of using gpt-4o-2024-08-06 with temperature and a top-p threshold equal to 0.5.

To test the final SDG pipeline, we chose an unbiased NER dataset in European Portuguese, considering the main goal of research. We also chose an unbiased PLM not only because it was recently made available, with a significant size, but also to test if utility disappeared with a different encoder.

## 6.2 Conclusions

Several conclusions and interesting results were achieved during the course of this dissertation. We found that GPT-4o was the best annotator considering the number of participants annotated and the NER performance gains. The alternative annotator was a UMLS-based tool that finds exact string matches between text spans in the health records and a biomedical vocabulary in Portuguese made available using a data source called UMLS.

One of the main results achieved, despite the usage of better health records to augment GPT-4o or of a system/user prompt structure, was the addition of a random disease or condition to the user prompt. Using it enabled us to achieve higher NER performance while reducing the risks of privacy disclosure. Despite the research question focusing on NER performance, privacy disclosure is an important concept we need to keep in mind in this line of work.

In terms of NER performance, the main conclusion was the quality level of the health reports generated and annotated. With them, we were able to achieve much closer NER performance between data shortage scenarios. More specifically, we were able to achieve similar NER performance using three times less real health records for training using the synthetic health records generated using our approach. We also outperformed baseline performance.

It was also important to test our approach using a different dataset and model architecture. By doing so, we were able to, once again, achieve higher NER performance, and thus outperforming the baselines.

## 6.3 Further Work

Considering the results achieved and their interpretation, there are several ways in which this work can be further developed.

First, although GPT-4o was chosen, considering it is a top performer in the industry with fast token generation times, other models could be considered cheaper to use, such as the Gemini family of LLMs. Furthermore, there is always the possibility to improve the prompt engineering process. The usage of more complex classification heads, such as ones with CRF layers, can

impact utility. it can be an interesting experiment measuring the utility of a sample of SEHRs across several architectures, not just one.

GPT-3.5 and GPT-4o, can generate and annotate SEHRs for sequence tagging tasks. An argument can be made that if such LLMs can generate and annotate synthetic data, they can also directly predict the STYs of a given EHR. Hence, in future works, one line of work to be considered is to measure baseline performance not only with a model fine-tuned without SEHRs but also with LLMs such as GPT-3.5 and GPT-4o. For this work, however, that was not relevant to measure the utility of the SEHRs. Still, it could be an interesting experiment.

Another line of work could be Relation Extraction. It could be interesting to see if a SDG pipeline similar to ours could generate relations between participants so that RE performance increases.

Despite not being shown in Chapter 5, an initial line of work was developed that uses wrong predictions to learn about the data and target specific problems. Despite the SDG pipeline positive results, a possible issue is that it generates SEHRs based on the random CONCEPT component. However, several of those random diseases or conditions might be irrelevant, thus generating SEHRs with little to no utility or added value. Therefore, the main idea is for the user to learn where the NER models are not able to predict correctly in the validation data. Knowing that, a SDG pipeline either at the sentence or document level could be used to try to generate samples with possibly more utility.

The preliminary work consisted of using unsupervised learning to cluster what we called "entity embeddings". These are not token embeddings neither sentence embeddings. We adapted the approach in the sentence embeddings paper to, instead, compute entity embeddings. We noticed that the PLM choice to compute the entity embeddings is crucial. Preliminary results showed that it needs to be the same PLM fine-tuned for the specific NER task. This way we can cluster entity embeddings that consider both its context window and the NER task. We used K-Means and chose the optimal number of clusters based on the highest silhouette score. Results showed similar or identical participants being clustered. This idea can even lower SDG costs by focusing on sentence-level augmentation. We also implemented in this new approach separation of wrong entities between partial, missed, and incorrect predictions. If a developer desires to improve recall, for example, then he/she could cluster missed predictions only, and then generate samples.



## Appendix A

# Appendix - SEHR examples and test result selection

In this appendix, a few examples of the SEHRs generated are shown. In this case, the SEHRs were the ones generated using gpt-4o-2024-08-06 in Section 5.4. Also, an example of a table is given to illustrate the explanation given for Figure 4.3. This table shows the results of the training pipeline diagram and how all the results shown in this dissertation were chosen.

### A.1 SEHR examples

In these SEHRs generated and annotated using the final SDG pipeline, we can see two main XML sections: TEXT and TAGS. TEXT consists of the actual synthetic text report, while TAGS are the annotations made.

We noticed that some of the text spans in the annotations were incorrectly generated (given the start and end components of the annotations). GPT-4o had punctual difficulties correctly giving the participant span. This was corrected with pre-processing using the "text" component of the annotation. The annotation was excluded when the "text" component and the respective annotation in TEXT didn't match (after doing some pre-processing such as lowercase). That being said, only a small percentage of annotations were excluded.

#### A.1.1 Example 1

```
<?xml version='1.0' encoding='UTF-8'?>
<ANNOTATIONS>
<TEXT>Paciente: Joao Silva, 32 anos, solteiro, natural de SP, estudante.

QP: Baixo peso em relacao a idade gestacional com sinais de ma nutricao fetal.
```

HMP: Paciente nasceu prematuro, com 32 semanas de gestacao, apresentando baixo peso ao nascer (1,5 kg). Desde entao, tem acompanhamento pediatrico regular devido a sinais de ma nutricao fetal.

S: Mae relata que Joao tem dificuldade para ganhar peso, mesmo com alimentacao adequada. Apresenta cansaco frequente e palidez.

O: Exame fisico: BEG, normocorado, hidratado, anicterico, acianotico. Peso : 15 kg (abaixo **do** percentil 3 para a idade). Altura: 95 cm (abaixo **do** percentil 3 para a idade). IMC: 16,6 kg/m<sup>2</sup>. PA: 90/60 mmHg. Pulso: 80 bpm. FR: 20 mrpm. Ausculta pulmonar: MV+ sem RA. Exame cardiovascular: RCR em 2T, sem sopros. Abdome: plano, flacido, indolor a palpacao, RHA presentes, sem visceromegalias. MMII: sem edemas.

Laboratoriais: Hemograma completo: Hb 11 g/dL, Ht 33%, Leucocitos 7.000/mm<sup>3</sup>, Plaquetas 250.000/mm<sup>3</sup>. Ferro serico: 50 g/dL. Ferritina: 15 ng/mL. Albumina: 3,5 g/dL. Proteinas totais: 6,0 g/dL. Glicose: 85 mg/dL. Ureia: 30 mg/dL. Creatinina: 0,5 mg/dL. Eletrolitos: Na 140 mmol/L, K 4,0 mmol/L, Ca 9,0 mg/dL.

A: Baixo peso em relacao a idade gestacional. Ma nutricao fetal.

P: Acompanhamento com nutricionista para orientacao dietetica. Suplementacao vitaminica e de ferro. Retorno em 3 meses para reavaliacao clinica e laboratorial.</TEXT>

<TAGS>

<annotation id="1" tag="Disease or Syndrome" start="0" end="5" text="Baixo peso" />

<annotation id="2" tag="Disease or Syndrome" start="6" end="32" text="idade gestacional" />

<annotation id="3" tag="Sign or Symptom" start="33" end="50" text="ma nutricao fetal" />

<annotation id="4" tag="Sign or Symptom" start="125" end="136" text="prematuro" />

<annotation id="5" tag="Quantitative Concept" start="157" end="163" text="32 semanas" />

<annotation id="6" tag="Quantitative Concept" start="191" end="196" text="1,5 kg" />

<annotation id="7" tag="Sign or Symptom" start="234" end="250" text="ma nutricao fetal" />

<annotation id="8" tag="Sign or Symptom" start="276" end="290" text="dificuldade para ganhar peso" />

<annotation id="9" tag="Sign or Symptom" start="315" end="322" text="cansaco" />

```
<annotation id="10" tag="Sign or Symptom" start="333" end="339" text="
palidez" />
<annotation id="11" tag="Sign or Symptom" start="370" end="373" text="BEG"
/>
<annotation id="12" tag="Sign or Symptom" start="375" end="386" text="
normocorado" />
<annotation id="13" tag="Sign or Symptom" start="388" end="397" text="
hidratado" />
<annotation id="14" tag="Sign or Symptom" start="399" end="408" text="
anicterico" />
<annotation id="15" tag="Sign or Symptom" start="410" end="419" text="
acianotico" />
<annotation id="16" tag="Quantitative Concept" start="428" end="433" text=
"15 kg" />
<annotation id="17" tag="Quantitative Concept" start="458" end="463" text=
"95 cm" />
<annotation id="18" tag="Quantitative Concept" start="490" end="500" text=
"16,6 kg/m2" />
<annotation id="19" tag="Quantitative Concept" start="507" end="518" text=
"90/60 mmHg" />
<annotation id="20" tag="Quantitative Concept" start="526" end="533" text=
"80 bpm" />
<annotation id="21" tag="Quantitative Concept" start="539" end="546" text=
"20 mrpm" />
<annotation id="22" tag="Sign or Symptom" start="563" end="566" text="RA"
/>
<annotation id="23" tag="Sign or Symptom" start="594" end="600" text="
sopros" />
<annotation id="24" tag="Sign or Symptom" start="627" end="634" text="
plano" />
<annotation id="25" tag="Sign or Symptom" start="636" end="643" text="
flacido" />
<annotation id="26" tag="Sign or Symptom" start="645" end="653" text="
indolor" />
<annotation id="27" tag="Sign or Symptom" start="671" end="686" text="
visceromegalias" />
<annotation id="28" tag="Sign or Symptom" start="693" end="699" text="
edemas" />
<annotation id="29" tag="Laboratory or Test Result" start="719" end="733"
text="Hemograma completo" />
<annotation id="30" tag="Laboratory or Test Result" start="735" end="746"
text="Hb 11 g/dL" />
<annotation id="31" tag="Laboratory or Test Result" start="748" end="755"
text="Ht 33%" />
```

```

<annotation id="32" tag="Laboratory or Test Result" start="757" end="774"
text="Leucocitos 7.000/mm3" />
<annotation id="33" tag="Laboratory or Test Result" start="776" end="795"
text="Plaquetas 250.000/mm3" />
<annotation id="34" tag="Laboratory or Test Result" start="797" end="812"
text="Ferro serico: 50 g/dL" />
<annotation id="35" tag="Laboratory or Test Result" start="814" end="832"
text="Ferritina: 15 ng/mL" />
<annotation id="36" tag="Laboratory or Test Result" start="834" end="850"
text="Albumina: 3,5 g/dL" />
<annotation id="37" tag="Laboratory or Test Result" start="852" end="874"
text="Proteinas totais: 6,0 g/dL" />
<annotation id="38" tag="Laboratory or Test Result" start="876" end="889"
text="Glicose: 85 mg/dL" />
<annotation id="39" tag="Laboratory or Test Result" start="891" end="903"
text="Ureia: 30 mg/dL" />
<annotation id="40" tag="Laboratory or Test Result" start="905" end="921"
text="Creatinina: 0,5 mg/dL" />
<annotation id="41" tag="Laboratory or Test Result" start="923" end="940"
text="Eletrolitos: Na 140 mmol/L" />
<annotation id="42" tag="Laboratory or Test Result" start="942" end="955"
text="K 4,0 mmol/L" />
<annotation id="43" tag="Laboratory or Test Result" start="957" end="968"
text="Ca 9,0 mg/dL" />
<annotation id="44" tag="Disease or Syndrome" start="970" end="1003" text=
"Baixo peso em relacao a idade gestacional" />
<annotation id="45" tag="Disease or Syndrome" start="1005" end="1021" text
="Ma nutricao fetal" />
<annotation id="46" tag="Diagnostic Procedure" start="1023" end="1037"
text="Acompanhamento" />
<annotation id="47" tag="Diagnostic Procedure" start="1039" end="1052"
text="Suplementacao" />
<annotation id="48" tag="Diagnostic Procedure" start="1054" end="1064"
text="Retorno" />
</TAGS>
</ANNOTATIONS>

```

### A.1.2 Example 2

```

<?xml version='1.0' encoding='UTF-8'?>
<ANNOTATIONS>
<TEXT>Paciente de 45 anos, sexo feminino, apresenta alternancia entre
constipacao e diarreia ha 6 meses. Relata episodios de dor abdominal

```

intensa, flatulencia e sensacao de inchaco. Nega febre, nauseas ou vomitos . Refere perda de peso de 5 kg no ultimo mes. Exames laboratoriais realizados em 15/09/2023: Hemograma completo: Hb 13,2 g/dL, Ht 39%, Leucocitos 6.000/mm3, Plaquetas 250.000/mm3. Bioquimica: Glicose 90 mg/dL, Creatinina 0,8 mg/dL, Ureia 30 mg/dL, Na 140 mmol/L, K 4,5 mmol/L. Exame de fezes: presenca de muco, ausencia de sangue. Colonoscopia realizada em 20/09/2023 revelou inflamacao leve no colon descendente. Diagnostico: Sindrome **do** Intestino Irritavel (SII). Paciente em uso de probioticos e dieta rica em fibras. Retorno em 3 meses para reavaliacao.</TEXT>

<TAGS>

<annotation id="1" tag="Sign or Symptom" start="47" end="59" text="constipacao" abbr=""/>

<annotation id="2" tag="Sign or Symptom" start="62" end="70" text="diarreia" abbr=""/>

<annotation id="3" tag="Quantitative Concept" start="74" end="82" text="6 meses" abbr=""/>

<annotation id="4" tag="Sign or Symptom" start="92" end="110" text="dor abdominal" abbr=""/>

<annotation id="5" tag="Sign or Symptom" start="119" end="130" text="flatulencia" abbr=""/>

<annotation id="6" tag="Sign or Symptom" start="134" end="150" text="sensacao de inchaco" abbr=""/>

<annotation id="7" tag="Sign or Symptom" start="158" end="163" text="febre" abbr=""/>

<annotation id="8" tag="Sign or Symptom" start="165" end="172" text="nauseas" abbr=""/>

<annotation id="9" tag="Sign or Symptom" start="176" end="183" text="vomitos" abbr=""/>

<annotation id="10" tag="Sign or Symptom" start="192" end="206" text="perda de peso" abbr=""/>

<annotation id="11" tag="Quantitative Concept" start="210" end="214" text="5 kg" abbr=""/>

<annotation id="12" tag="Quantitative Concept" start="218" end="230" text="ultimo mes" abbr=""/>

<annotation id="13" tag="Diagnostic Procedure" start="239" end="258" text="Exames laboratoriais" abbr=""/>

<annotation id="14" tag="Quantitative Concept" start="274" end="276" text="15" abbr=""/>

<annotation id="15" tag="Quantitative Concept" start="277" end="279" text="09" abbr=""/>

<annotation id="16" tag="Quantitative Concept" start="280" end="284" text="2023" abbr=""/>

<annotation id="17" tag="Laboratory or Test Result" start="286" end="303" text="Hemograma completo" abbr=""/>

```

<annotation id="18" tag="Laboratory or Test Result" start="305" end="314"
text="Hb 13,2 g/dL" abbr=""/>
<annotation id="19" tag="Laboratory or Test Result" start="316" end="323"
text="Ht 39%" abbr=""/>
<annotation id="20" tag="Laboratory or Test Result" start="325" end="344"
text="Leucocitos 6.000/mm3" abbr=""/>
<annotation id="21" tag="Laboratory or Test Result" start="346" end="366"
text="Plaquetas 250.000/mm3" abbr=""/>
<annotation id="22" tag="Laboratory or Test Result" start="368" end="378"
text="Bioquimica" abbr=""/>
<annotation id="23" tag="Laboratory or Test Result" start="380" end="393"
text="Glicose 90 mg/dL" abbr=""/>
<annotation id="24" tag="Laboratory or Test Result" start="395" end="410"
text="Creatinina 0,8 mg/dL" abbr=""/>
<annotation id="25" tag="Laboratory or Test Result" start="412" end="423"
text="Ureia 30 mg/dL" abbr=""/>
<annotation id="26" tag="Laboratory or Test Result" start="425" end="436"
text="Na 140 mmol/L" abbr=""/>
<annotation id="27" tag="Laboratory or Test Result" start="438" end="448"
text="K 4,5 mmol/L" abbr=""/>
<annotation id="28" tag="Diagnostic Procedure" start="450" end="464" text=
"Exame de fezes" abbr=""/>
<annotation id="29" tag="Laboratory or Test Result" start="466" end="479"
text="presenca de muco" abbr=""/>
<annotation id="30" tag="Laboratory or Test Result" start="481" end="498"
text="ausencia de sangue" abbr=""/>
<annotation id="31" tag="Diagnostic Procedure" start="500" end="511" text=
"Colonoscopia" abbr=""/>
<annotation id="32" tag="Quantitative Concept" start="523" end="533" text=
"20/09/2023" abbr=""/>
<annotation id="33" tag="Laboratory or Test Result" start="535" end="558"
text="inflamacao leve no colon descendente" abbr=""/>
<annotation id="34" tag="Disease or Syndrome" start="572" end="599" text="
Sindrome do Intestino Irritavel" abbr="SII"/>
<annotation id="35" tag="Diagnostic Procedure" start="601" end="605" text=
"SII" abbr=""/>
<annotation id="36" tag="Sign or Symptom" start="607" end="615" text="
Paciente" abbr=""/>
<annotation id="37" tag="Sign or Symptom" start="617" end="621" text="uso"
abbr=""/>
<annotation id="38" tag="Sign or Symptom" start="623" end="634" text="
probioticos" abbr=""/>
<annotation id="39" tag="Sign or Symptom" start="638" end="652" text="
dieta rica em fibras" abbr=""/>

```

```
<annotation id="40" tag="Quantitative Concept" start="654" end="663" text=
"3 meses" abbr=""/>
</TAGS>
</ANNOTATIONS>
```

### A.1.3 Example 3

```
<?xml version='1.0' encoding='UTF-8'?>
<ANNOTATIONS>
<TEXT>Paciente de 45 anos, masculino, com historico de hipertensao
arterial (HAS) e diabetes mellitus tipo 2 (DM2). Relata dor intensa e
inchaco no penis apos trauma direto durante atividade fisica. Nega febre,
calafrios ou outras queixas sistemicas.

Ao exame fisico:
- PA: 130/85 mmHg
- FC: 75 bpm
- FR: 18 irpm
- Temperatura: 36,7C
- BEG, hidratado, corado, afebril.
- Abdome: plano, flacido, indolor a palpacao, RHA presentes.
- MMII: sem edemas, pulsos perifericos presentes e simetricos.
- Genitalia: presenca de hematoma extenso em penis, sem sinais de infeccao
ou necrose.

Exames laboratoriais:
- Hemograma: Hb 14,2 g/dL, Ht 42%, Leucocitos 7.500/mm3, Plaquetas
250.000/mm3
- Glicemia de jejum: 110 mg/dL
- Creatinina: 1,0 mg/dL
- Ureia: 30 mg/dL
- Potassio: 4,5 mmol/L
- Sodio: 140 mmol/L

Exames de imagem:
- Ultrassonografia (US) de penis: hematoma subcutaneo extenso, sem sinais
de fratura peniana ou lesao uretral.

Diagnostico:
- Hematoma peniano
- HAS
- DM2
```

## Conduta:

- Manter observacao e repouso.
- Aplicacao de compressas frias no local.
- Analgesia com paracetamol 500 mg, 1 cp a cada 6 horas, se necessario.
- Retorno em 7 dias para reavaliacao.

## Medicacoes em uso:

- Losartana 50 mg, 1 cp ao dia
- Metformina 850 mg, 1 cp 12/12h
- Paracetamol 500 mg, 1 cp a cada 6 horas, se necessario

## Orientacoes:

- Evitar atividades fisicas intensas ate a completa resolucao **do** hematoma.
- Manter hidratacao adequada.
- Monitorar sinais de infeccao (febre, aumento da dor, secrecao purulenta)
- .
- Retorno imediato ao servico de saude em caso de piora dos sintomas.</

TEXT&gt;

&lt;TAGS&gt;

```

<annotation id="1" tag="Disease or Syndrome" start="49" end="72" text="
hipertensao arterial" abbr="HAS" />
<annotation id="2" tag="Disease or Syndrome" start="76" end="99" text="
diabetes mellitus tipo 2" abbr="DM2" />
<annotation id="3" tag="Sign or Symptom" start="109" end="117" text="dor
intensa" />
<annotation id="4" tag="Sign or Symptom" start="120" end="126" text="
inchaco" />
<annotation id="5" tag="Sign or Symptom" start="131" end="136" text="penis
" />
<annotation id="6" tag="Sign or Symptom" start="143" end="148" text="
trauma" />
<annotation id="7" tag="Sign or Symptom" start="162" end="167" text="febre
" />
<annotation id="8" tag="Sign or Symptom" start="169" end="178" text="
calafrios" />
<annotation id="9" tag="Sign or Symptom" start="182" end="190" text="
queixas" />
<annotation id="10" tag="Quantitative Concept" start="211" end="220" text=
"130/85 mmHg" />
<annotation id="11" tag="Quantitative Concept" start="228" end="235" text=
"75 bpm" />
<annotation id="12" tag="Quantitative Concept" start="243" end="251" text=
"18 irpm" />
<annotation id="13" tag="Quantitative Concept" start="266" end="272" text=
"36,7C" />

```

```
<annotation id="14" tag="Sign or Symptom" start="275" end="278" text="BEG"
/>
<annotation id="15" tag="Sign or Symptom" start="280" end="289" text="
hidratado" />
<annotation id="16" tag="Sign or Symptom" start="291" end="297" text="
corado" />
<annotation id="17" tag="Sign or Symptom" start="299" end="306" text="
afebril" />
<annotation id="18" tag="Sign or Symptom" start="309" end="314" text="
Abdome" />
<annotation id="19" tag="Sign or Symptom" start="316" end="321" text="
plano" />
<annotation id="20" tag="Sign or Symptom" start="323" end="329" text="
flacido" />
<annotation id="21" tag="Sign or Symptom" start="331" end="338" text="
indolor" />
<annotation id="22" tag="Sign or Symptom" start="342" end="354" text="RHA
presentes" />
<annotation id="23" tag="Sign or Symptom" start="357" end="361" text="MMII
" />
<annotation id="24" tag="Sign or Symptom" start="367" end="376" text="sem
edemas" />
<annotation id="25" tag="Sign or Symptom" start="378" end="403" text="
pulsos perifericos presentes" />
<annotation id="26" tag="Sign or Symptom" start="408" end="417" text="
Genitalia" />
<annotation id="27" tag="Sign or Symptom" start="420" end="428" text="
hematoma" />
<annotation id="28" tag="Sign or Symptom" start="429" end="437" text="
extenso" />
<annotation id="29" tag="Sign or Symptom" start="441" end="446" text="
penis" />
<annotation id="30" tag="Sign or Symptom" start="448" end="454" text="
sinais" />
<annotation id="31" tag="Sign or Symptom" start="458" end="467" text="
infeccao" />
<annotation id="32" tag="Sign or Symptom" start="471" end="478" text="
necrose" />
<annotation id="33" tag="Laboratory or Test Result" start="500" end="513"
text="Hemograma: Hb 14,2 g/dL" />
<annotation id="34" tag="Laboratory or Test Result" start="515" end="520"
text="Ht 42%" />
<annotation id="35" tag="Laboratory or Test Result" start="522" end="537"
text="Leuccitos 7.500/mm3" />
```

```
<annotation id="36" tag="Laboratory or Test Result" start="539" end="558"
text="Plaquetas 250.000/mm3" />
<annotation id="37" tag="Laboratory or Test Result" start="560" end="579"
text="Glicemia de jejum: 110 mg/dL" />
<annotation id="38" tag="Laboratory or Test Result" start="581" end="594"
text="Creatinina: 1,0 mg/dL" />
<annotation id="39" tag="Laboratory or Test Result" start="596" end="606"
text="Ureia: 30 mg/dL" />
<annotation id="40" tag="Laboratory or Test Result" start="608" end="624"
text="Potassio: 4,5 mmol/L" />
<annotation id="41" tag="Laboratory or Test Result" start="626" end="641"
text="Sodio: 140 mmol/L" />
<annotation id="42" tag="Diagnostic Procedure" start="663" end="689" text=
"Ultrassonografia (US)" />
<annotation id="43" tag="Sign or Symptom" start="693" end="698" text="
penis" />
<annotation id="44" tag="Sign or Symptom" start="700" end="708" text="
hematoma" />
<annotation id="45" tag="Sign or Symptom" start="710" end="721" text="
subcutaneo" />
<annotation id="46" tag="Sign or Symptom" start="723" end="730" text="
extenso" />
<annotation id="47" tag="Sign or Symptom" start="736" end="743" text="
fratura" />
<annotation id="48" tag="Sign or Symptom" start="744" end="751" text="
peniana" />
<annotation id="49" tag="Sign or Symptom" start="755" end="761" text="
lesao" />
<annotation id="50" tag="Sign or Symptom" start="762" end="769" text="
uretral" />
<annotation id="51" tag="Disease or Syndrome" start="783" end="799" text="
Hematoma peniano" />
<annotation id="52" tag="Disease or Syndrome" start="802" end="805" text="
HAS" />
<annotation id="53" tag="Disease or Syndrome" start="808" end="811" text="
DM2" />
<annotation id="54" tag="Diagnostic Procedure" start="835" end="846" text=
"observacao" />
<annotation id="55" tag="Diagnostic Procedure" start="849" end="856" text=
"repouso" />
<annotation id="56" tag="Diagnostic Procedure" start="860" end="883" text=
"compressas frias no local" />
<annotation id="57" tag="Diagnostic Procedure" start="887" end="895" text=
"Analgesia" />
```

```

<annotation id="58" tag="Quantitative Concept" start="900" end="911" text=
"500 mg" />
<annotation id="59" tag="Quantitative Concept" start="913" end="927" text=
"1 cp a cada 6 horas" />
<annotation id="60" tag="Diagnostic Procedure" start="930" end="938" text=
"reavaliacao" />
<annotation id="61" tag="Quantitative Concept" start="1002" end="1013"
text="50 mg" />
<annotation id="62" tag="Quantitative Concept" start="1020" end="1031"
text="850 mg" />
<annotation id="63" tag="Quantitative Concept" start="1033" end="1047"
text="1 cp 12/12h" />
<annotation id="64" tag="Quantitative Concept" start="1050" end="1061"
text="500 mg" />
<annotation id="65" tag="Quantitative Concept" start="1063" end="1077"
text="1 cp a cada 6 horas" />
<annotation id="66" tag="Sign or Symptom" start="1123" end="1130" text="
hidratacao" />
<annotation id="67" tag="Sign or Symptom" start="1144" end="1153" text="
infeccao" />
<annotation id="68" tag="Sign or Symptom" start="1155" end="1160" text="
febre" />
<annotation id="69" tag="Sign or Symptom" start="1162" end="1176" text="
aumento da dor" />
<annotation id="70" tag="Sign or Symptom" start="1178" end="1196" text="
secrecao purulenta" />
<annotation id="71" tag="Sign or Symptom" start="1204" end="1210" text="
piora" />
<annotation id="72" tag="Sign or Symptom" start="1214" end="1222" text="
sintomas" />
</TAGS>
</ANNOTATIONS>

```

## A.2 Test result selection

In this section, a total of three examples are given to illustrate how the results of the diagram shown in Figure 4.3. In bold are shown the validation and test results chosen. This choice was made, as mentioned, considering the highest strict F1 score on the validation data.

### A.2.1 Example 1

The test results in bold can be seen in Table 5.30 for gpt-4o-2024-08-06. The highest strict F1 score on the validation data occurred in restart 3 using the last epoch criterion, with 38.11%.

| Restart      | Model instance selection criteria | Data set    | Metric | Strict        | Partial       | Type          | Loss          |
|--------------|-----------------------------------|-------------|--------|---------------|---------------|---------------|---------------|
| Restart<br>1 | Last epoch criterion              | Development | P      | 30.21%        | 38.02%        | 45.83%        | 0.0099        |
|              |                                   |             | R      | 49.10%        | 61.79%        | 74.49%        |               |
|              |                                   |             | F1     | 37.40%        | 47.08%        | 56.75%        |               |
|              |                                   | Test        | P      | 31.56%        | 39.32%        | 47.08%        | 0.0103        |
|              |                                   |             | R      | 49.80%        | 62.05%        | 74.30%        |               |
|              |                                   |             | F1     | 38.63%        | 48.14%        | 57.64%        |               |
|              | Lowest overfit criterion          | Development | P      | 30.21%        | 38.02%        | 45.83%        | 0.0099        |
|              |                                   |             | R      | 49.10%        | 61.79%        | 74.49%        |               |
|              |                                   |             | F1     | 37.40%        | 47.08%        | 56.75%        |               |
|              |                                   | Test        | P      | 31.56%        | 39.32%        | 47.08%        | 0.0103        |
|              |                                   |             | R      | 49.80%        | 62.05%        | 74.30%        |               |
|              |                                   |             | F1     | 38.63%        | 48.14%        | 57.64%        |               |
| Restart<br>2 | Last epoch criterion              | Development | P      | 29.40%        | 37.81%        | 46.23%        | 0.0100        |
|              |                                   |             | R      | 47.52%        | 61.12%        | 74.72%        |               |
|              |                                   |             | F1     | 36.32%        | 46.72%        | 57.12%        |               |
|              |                                   | Test        | P      | 31.78%        | 39.48%        | 47.19%        | 0.0104        |
|              |                                   |             | R      | 49.70%        | 61.75%        | 73.80%        |               |
|              |                                   |             | F1     | 38.77%        | 48.17%        | 57.57%        |               |
|              | Lowest overfit criterion          | Development | P      | 29.40%        | 37.81%        | 46.23%        | 0.0100        |
|              |                                   |             | R      | 47.52%        | 61.12%        | 74.72%        |               |
|              |                                   |             | F1     | 36.32%        | 46.72%        | 57.12%        |               |
|              |                                   | Test        | P      | 31.78%        | 39.48%        | 47.19%        | 0.0104        |
|              |                                   |             | R      | 49.70%        | 61.75%        | 73.80%        |               |
|              |                                   |             | F1     | 38.77%        | 48.17%        | 57.57%        |               |
| Restart<br>3 | Last epoch criterion              | Development | P      | <b>31.71%</b> | <b>40.37%</b> | <b>49.03%</b> | <b>0.0095</b> |
|              |                                   |             | R      | <b>47.74%</b> | <b>60.78%</b> | <b>73.81%</b> |               |
|              |                                   |             | F1     | <b>38.11%</b> | <b>48.51%</b> | <b>58.92%</b> |               |
|              |                                   | Test        | P      | <b>34.32%</b> | <b>42.74%</b> | <b>51.16%</b> | <b>0.0099</b> |
|              |                                   |             | R      | <b>48.70%</b> | <b>60.65%</b> | <b>72.60%</b> |               |
|              |                                   |             | F1     | <b>40.26%</b> | <b>50.14%</b> | <b>60.02%</b> |               |
|              | Lowest overfit criterion          | Development | P      | 31.71%        | 40.37%        | 49.03%        | 0.0095        |
|              |                                   |             | R      | 47.74%        | 60.78%        | 73.81%        |               |
|              |                                   |             | F1     | 38.11%        | 48.51%        | 58.92%        |               |
|              |                                   | Test        | P      | 34.32%        | 42.74%        | 51.16%        | 0.0099        |
|              |                                   |             | R      | 48.70%        | 60.65%        | 72.60%        |               |
|              |                                   |             | F1     | 40.26%        | 50.14%        | 60.02%        |               |

Bold values indicate the results from the model instance chosen to compare against other TRSTR schemes and baseline results, considering the lowest overfit criteria.

Table A.1: Results of BioBERTpt(all) fine-tuned for the Sign or Symptom STY with an augmented training corpus of 100 real EHRs with 300 SEHRs generated and annotated using gpt-4o-2024-08-06

### A.2.2 Example 2

The test results in bold can be seen in Table 5.40 for gpt-4o-2024-08-06. The highest strict F1 score on the validation data occurred in restart 3 using the last epoch criterion, with 63.31%.

| Restart   | Model instance selection criteria | Data set    | Metric | Strict        | Partial       | Type          | Loss          |
|-----------|-----------------------------------|-------------|--------|---------------|---------------|---------------|---------------|
| Restart 1 | Last epoch criterion              | Development | P      | 54.70%        | 59.29%        | 63.89%        | 0.0021        |
|           |                                   |             | R      | 70.52%        | 76.45%        | 82.37%        |               |
|           |                                   |             | F1     | 61.61%        | 66.79%        | 71.96%        |               |
|           |                                   | Test        | P      | 58.76%        | 63.15%        | 67.53%        | 0.0021        |
|           |                                   |             | R      | 74.68%        | 80.25%        | 85.82%        |               |
|           |                                   |             | F1     | 65.77%        | 70.68%        | 75.59%        |               |
|           | Lowest overfit criterion          | Development | P      | 54.70%        | 59.29%        | 63.89%        | 0.0021        |
|           |                                   |             | R      | 70.52%        | 76.45%        | 82.37%        |               |
|           |                                   |             | F1     | 61.61%        | 66.79%        | 71.96%        |               |
| Test      |                                   | P           | 58.76% | 63.15%        | 67.53%        | 0.0021        |               |
|           |                                   | R           | 74.68% | 80.25%        | 85.82%        |               |               |
|           |                                   | F1          | 65.77% | 70.68%        | 75.59%        |               |               |
| Restart 2 | Last epoch criterion              | Development | P      | 57.27%        | 61.59%        | 65.91%        | 0.0022        |
|           |                                   |             | R      | 69.42%        | 74.66%        | 79.89%        |               |
|           |                                   |             | F1     | 62.76%        | 67.50%        | 72.23%        |               |
|           |                                   | Test        | P      | 59.09%        | 62.91%        | 66.74%        | 0.0022        |
|           |                                   |             | R      | 72.41%        | 77.09%        | 81.77%        |               |
|           |                                   |             | F1     | 65.07%        | 69.28%        | 73.49%        |               |
|           | Lowest overfit criterion          | Development | P      | 57.27%        | 61.59%        | 65.91%        | 0.0022        |
|           |                                   |             | R      | 69.42%        | 74.66%        | 79.89%        |               |
|           |                                   |             | F1     | 62.76%        | 67.50%        | 72.23%        |               |
| Test      |                                   | P           | 59.09% | 62.91%        | 66.74%        | 0.0022        |               |
|           |                                   | R           | 72.41% | 77.09%        | 81.77%        |               |               |
|           |                                   | F1          | 65.07% | 69.28%        | 73.49%        |               |               |
| Restart 3 | Last epoch criterion              | Development | P      | <b>59.61%</b> | <b>64.36%</b> | <b>69.10%</b> | <b>0.0021</b> |
|           |                                   |             | R      | <b>67.49%</b> | <b>72.87%</b> | <b>78.24%</b> |               |
|           |                                   |             | F1     | <b>63.31%</b> | <b>68.35%</b> | <b>73.39%</b> |               |
|           |                                   | Test        | P      | <b>63.97%</b> | <b>67.90%</b> | <b>71.83%</b> | <b>0.0020</b> |
|           |                                   |             | R      | <b>74.18%</b> | <b>78.73%</b> | <b>83.29%</b> |               |
|           |                                   |             | F1     | <b>68.70%</b> | <b>72.92%</b> | <b>77.14%</b> |               |
|           | Lowest overfit criterion          | Development | P      | 59.61%        | 64.36%        | 69.10%        | 0.0021        |
|           |                                   |             | R      | 67.49%        | 72.87%        | 78.24%        |               |
|           |                                   |             | F1     | 63.31%        | 68.35%        | 73.39%        |               |
| Test      |                                   | P           | 63.97% | 67.90%        | 71.83%        | 0.0020        |               |
|           |                                   | R           | 74.18% | 78.73%        | 83.29%        |               |               |
|           |                                   | F1          | 68.70% | 72.92%        | 77.14%        |               |               |

Bold values indicate the results from the model instance chosen to compare against other TRSTR schemes and baseline results, considering the lowest overfit criteria.

Table A.2: Results of BioBERTpt(all) fine-tuned for the Diagnostic Procedure STY with an augmented training corpus of 600 real EHRs with 300 SEHRs generated and annotated using gpt-4o-2024-08-06

### A.2.3 Example 3

The test results in bold can be seen in Table 5.36 for gpt-4o-2024-05-13. The highest strict F1 score on the validation data occurred in restart 2 using the last epoch criterion, with 42.37%.

| Restart      | Model instance selection criteria | Data set    | Metric | Strict        | Partial       | Type          | Loss          |
|--------------|-----------------------------------|-------------|--------|---------------|---------------|---------------|---------------|
| Restart<br>1 | Last epoch criterion              | Development | P      | 35.63%        | 47.46%        | 59.28%        | 0.0072        |
|              |                                   |             | R      | 46.76%        | 62.28%        | 77.80%        |               |
|              |                                   |             | F1     | 40.44%        | 53.87%        | 67.29%        |               |
|              |                                   | Test        | P      | 33.20%        | 45.65%        | 58.11%        | 0.0078        |
|              |                                   |             | R      | 42.60%        | 58.57%        | 74.54%        |               |
|              |                                   |             | F1     | 37.32%        | 51.31%        | 65.31%        |               |
|              | Lowest overfit criterion          | Development | P      | 35.63%        | 47.46%        | 59.28%        | 0.0072        |
|              |                                   |             | R      | 46.76%        | 62.28%        | 77.80%        |               |
|              |                                   |             | F1     | 40.44%        | 53.87%        | 67.29%        |               |
|              |                                   | Test        | P      | 33.20%        | 45.65%        | 58.11%        | 0.0078        |
|              |                                   |             | R      | 42.60%        | 58.57%        | 74.54%        |               |
|              |                                   |             | F1     | 37.32%        | 51.31%        | 65.31%        |               |
| Restart<br>2 | Last epoch criterion              | Development | P      | <b>38.46%</b> | <b>50.72%</b> | <b>62.98%</b> | <b>0.0071</b> |
|              |                                   |             | R      | <b>47.15%</b> | <b>62.18%</b> | <b>77.21%</b> |               |
|              |                                   |             | F1     | <b>42.37%</b> | <b>55.87%</b> | <b>69.37%</b> |               |
|              |                                   | Test        | P      | <b>34.79%</b> | <b>47.39%</b> | <b>60.00%</b> | <b>0.0083</b> |
|              |                                   |             | R      | <b>41.10%</b> | <b>55.99%</b> | <b>70.88%</b> |               |
|              |                                   |             | F1     | <b>37.68%</b> | <b>51.33%</b> | <b>64.99%</b> |               |
|              | Lowest overfit criterion          | Development | P      | 38.46%        | 50.72%        | 62.98%        | 0.0071        |
|              |                                   |             | R      | 47.15%        | 62.18%        | 77.21%        |               |
|              |                                   |             | F1     | 42.37%        | 55.87%        | 69.37%        |               |
|              |                                   | Test        | P      | 34.79%        | 47.39%        | 60.00%        | 0.0083        |
|              |                                   |             | R      | 41.10%        | 55.99%        | 70.88%        |               |
|              |                                   |             | F1     | 37.68%        | 51.33%        | 64.99%        |               |
| Restart<br>3 | Last epoch criterion              | Development | P      | 35.65%        | 47.43%        | 59.21%        | 0.0075        |
|              |                                   |             | R      | 46.37%        | 61.69%        | 77.01%        |               |
|              |                                   |             | F1     | 40.31%        | 53.63%        | 66.95%        |               |
|              |                                   | Test        | P      | 34.71%        | 47.06%        | 59.41%        | 0.0082        |
|              |                                   |             | R      | 43.26%        | 58.65%        | 74.04%        |               |
|              |                                   |             | F1     | 38.52%        | 52.22%        | 65.93%        |               |
|              | Lowest overfit criterion          | Development | P      | 35.65%        | 47.43%        | 59.21%        | 0.0075        |
|              |                                   |             | R      | 46.37%        | 61.69%        | 77.01%        |               |
|              |                                   |             | F1     | 40.31%        | 53.63%        | 66.95%        |               |
|              |                                   | Test        | P      | 34.71%        | 47.06%        | 59.41%        | 0.0082        |
|              |                                   |             | R      | 43.26%        | 58.65%        | 74.04%        |               |
|              |                                   |             | F1     | 38.52%        | 52.22%        | 65.93%        |               |

Bold values indicate the results from the model instance chosen to compare against other TRSTR schemes and baseline results, considering the lowest overfit criteria.

Table A.3: Results of BioBERTpt(all) fine-tuned for the Laboratory or Test Results STY with an augmented training corpus of 100 real EHRs with 300 SEHRs generated and annotated using gpt-4o-2024-05-13

# References

- General data protection regulation (gdpr) – official legal text. <https://gdpr-info.eu/>, 2021. Accessed: February 10, 2024.
- David H. Ackley, Geoffrey E. Hinton, and Terrence J. Sejnowski. A learning algorithm for boltzmann machines. *Cognitive Science*, 9(1):147–169, 1985. ISSN 0364-0213. doi: [https://doi.org/10.1016/S0364-0213\(85\)80012-4](https://doi.org/10.1016/S0364-0213(85)80012-4). URL <https://www.sciencedirect.com/science/article/pii/S0364021385800124>.
- James Allan, Jaime G. Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic Detection and Tracking Pilot Study Final Report. 1 1998. doi: 10.1184/R1/6626252.v1. URL [https://kilthub.cmu.edu/articles/journal\\_contribution/Topic\\_Detection\\_and\\_Tracking\\_Pilot\\_Study\\_Final\\_Report/6626252](https://kilthub.cmu.edu/articles/journal_contribution/Topic_Detection_and_Tracking_Pilot_Study_Final_Report/6626252).
- Nasser Alshammari and Saad Alanazi. The impact of using different annotation schemes on named entity recognition. *Egyptian Informatics Journal*, 22(3):295–302, 2021. ISSN 1110-8665. doi: 10.1016/j.eij.2020.10.004. URL <https://www.sciencedirect.com/science/article/pii/S1110866520301596>.
- M Ashburner, C A Ball, J A Blake, D Botstein, H Butler, J M Cherry, A P Davis, K Dolinski, S S Dwight, J T Eppig, M A Harris, D P Hill, L Issel-Tarver, A Kasarskis, S Lewis, J C Matese, J E Richardson, M Ringwald, G M Rubin, and G Sherlock. Gene ontology: tool for the unification of biology. the gene ontology consortium. *Nat Genet*, 25(1):25–29, May 2000.
- Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E. Hinton. Layer normalization, 2016.
- Edmon Begoli, Kris Brown, Sudarshan Srinivas, and Suzanne Tamang. Synthnotes: A generator framework for high-volume, high-fidelity synthetic mental health notes. In *2018 IEEE International Conference on Big Data (Big Data)*, pages 951–958, Seattle, WA, USA, 2018. doi: 10.1109/BigData.2018.8621981.
- Toms Bergmanis and Sharon Goldwater. Context sensitive neural lemmatization with Lematus. In Marilyn Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1391–1400, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. doi: 10.18653/v1/N18-1126. URL <https://aclanthology.org/N18-1126>.
- Steven Bird and Edward Loper. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL <https://aclanthology.org/P04-3031>.

- Sidney Black, Stella Biderman, Eric Hallahan, Quentin Anthony, Leo Gao, Laurence Golding, Horace He, Connor Leahy, Kyle McDonell, Jason Phang, Michael Pieler, Usven Sai Prashanth, Shivanshu Purohit, Laria Reynolds, Jonathan Tow, Ben Wang, and Samuel Weinbach. GPT-NeoX-20B: An open-source autoregressive language model. In Angela Fan, Suzana Ilic, Thomas Wolf, and Matthias Gallé, editors, *Proceedings of BigScience Episode #5 – Workshop on Challenges & Perspectives in Creating Large Language Models*, pages 95–136, virtual+Dublin, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.bigscience-1.9. URL <https://aclanthology.org/2022.bigscience-1.9>.
- Bernd Bohnet, Ryan McDonald, Gonalo Simões, Daniel Andor, Emily Pitler, and Joshua Maynez. Morphosyntactic tagging with a meta-BiLSTM model over context sensitive token encodings. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2642–2652, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1246. URL <https://aclanthology.org/P18-1246>.
- Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020. URL [https://proceedings.neurips.cc/paper\\_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2020/file/1457c0d6bfc4967418bfb8ac142f64a-Paper.pdf).
- Nathanael Chambers. NavyTime: Event and time ordering from raw text. In Suresh Manandhar and Deniz Yuret, editors, *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 73–77, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL <https://aclanthology.org/S13-2012>.
- Nathanael Chambers and Dan Jurafsky. Unsupervised learning of narrative event chains. In Johanna D. Moore, Simone Teufel, James Allan, and Sadaoki Furui, editors, *Proceedings of ACL-08: HLT*, pages 789–797, Columbus, Ohio, June 2008. Association for Computational Linguistics. URL <https://aclanthology.org/P08-1090>.
- Angel X. Chang and Christopher Manning. SUTime: A library for recognizing and normalizing time expressions. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation (LREC’12)*, pages 3735–3740, Istanbul, Turkey, May 2012. European Language Resources Association (ELRA). URL [http://www.lrec-conf.org/proceedings/lrec2012/pdf/284\\_Paper.pdf](http://www.lrec-conf.org/proceedings/lrec2012/pdf/284_Paper.pdf).
- Banghao Chen, Zhaofeng Zhang, Nicolas Langrené, and Shengxin Zhu. Unleashing the potential of prompt engineering in large language models: a comprehensive review, 2023.
- Alebachew Chiche and Betselot Yitagesu. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9(1):10, January 2022.

- Paul F Christiano, Jan Leike, Tom Brown, Miljan Martic, Shane Legg, and Dario Amodei. Deep reinforcement learning from human preferences. In I. Guyon, U. Von Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, volume 30. Curran Associates, Inc., 2017. URL [https://proceedings.neurips.cc/paper\\_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf](https://proceedings.neurips.cc/paper_files/paper/2017/file/d5e2c0adad503c91f91df240d0cd4e49-Paper.pdf).
- Yao-Shun Chuang, Chun-Teh Lee, Ryan Brandon, Trung Duong Tran, Oluwabunmi Tokede, Muhammad F. Walji, and Xiaoqian Jiang. Extracting periodontitis diagnosis in clinical notes with roberta and regular expression. In *2023 IEEE 11th International Conference on Healthcare Informatics (ICHI)*, pages 200–205, Houston, TX, USA, 2023. doi: 10.1109/ICHI57859.2023.00036.
- Haixing Dai, Zheng Liu, Wenxiong Liao, Xiaoke Huang, Zihao Wu, Lin Zhao, Wei Liu, Ninghao Liu, Sheng Li, Dajiang Zhu, Hongmin Cai, Quanzheng Li, Dinggang Shen, Tianming Liu, and Xiang Li. Chataug: Leveraging chatgpt for text data augmentation. *ArXiv*, abs/2302.13007, 2023. doi: 10.48550/arXiv.2302.13007.
- Marie-Catherine de Marneffe, Christopher D. Manning, Joakim Nivre, and Daniel Zeman. Universal Dependencies. *Computational Linguistics*, 47(2):255–308, June 2021. doi: 10.1162/coli\_a\_00402. URL <https://aclanthology.org/2021.cl-2.11>.
- João Vitor de Souza, Yohan Gumiel, Lucas Emanuel Oliveira, and Claudia Maria Moro. Named entity recognition for clinical portuguese corpus with conditional random fields and semantic groups. In *Anais do XIX Simpósio Brasileiro de Computação Aplicada à Saúde*, pages 318–323, Porto Alegre, RS, Brasil, 2019. SBC. doi: 10.5753/sbcas.2019.6269. URL <https://sol.sbc.org.br/index.php/sbcas/article/view/6269>.
- Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Tamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi: 10.18653/v1/N19-1423. URL <https://aclanthology.org/N19-1423>.
- H. Eyre, A. B. Chapman, K. S. Peterson, J. Shi, P. R. Alba, M. M. Jones, T. L. Box, S. L. DuVall, and O. V. Patterson. Launching into clinical space with medspaCy: a new clinical text processing toolkit in Python. *AMIA Annu Symp Proc*, 2021:438–447, 2021.
- Angela Fan, Mike Lewis, and Yann Dauphin. Hierarchical neural story generation. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 889–898, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1082. URL <https://aclanthology.org/P18-1082>.
- João Ferreira, Hugo Gonalo Oliveira, and Ricardo Rodrigues. Improving NLTK for Processing Portuguese. In Ricardo Rodrigues, Jan Janoušek, Luís Ferreira, Luísa Coheur, Fernando Batista, and Hugo Gonalo Oliveira, editors, *8th Symposium on Languages, Applications and Technologies (SLATE 2019)*, volume 74 of *Open Access Series in Informatics (OASICS)*,

- pages 18:1–18:9, Dagstuhl, Germany, 2019. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-114-6. doi: 10.4230/OASICS.SLATE.2019.18. URL <https://drops.dagstuhl.de/entities/document/10.4230/OASICS.SLATE.2019.18>.
- Lisa Ferro, Inderjeet Mani, Beth Sundheim, and George Wilson. TIDES Temporal Annotation Guidelines—Version 1.0.2. Technical report, The MITRE Corporation, 2001. Version 1.0.2.
- Lisa Ferro, Laurie Gerber, Inderjeet Mani, Beth Sundheim, and George Wilson. TIDES 2005 Standard for the Annotation of Temporal Expressions. Technical report, MITRE, 2005.
- Johann Frei and Frank Kramer. Annotated dataset creation through large language models for non-english medical nlp. *Journal of Biomedical Informatics*, 145:104478, 2023. ISSN 1532-0464. doi: <https://doi.org/10.1016/j.jbi.2023.104478>. URL <https://www.sciencedirect.com/science/article/pii/S1532046423001995>.
- Markus Freitag and Yaser Al-Onaizan. Beam search strategies for neural machine translation. In Thang Luong, Alexandra Birch, Graham Neubig, and Andrew Finch, editors, *Proceedings of the First Workshop on Neural Machine Translation*, pages 56–60, Vancouver, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-3207. URL <https://aclanthology.org/W17-3207>.
- Shang Gao, Olivera Kotevska, Alexandre Sorokine, and J Blair Christian. A pre-training and self-training approach for biomedical named entity recognition. *PLoS One*, 16(2):e0246310, February 2021.
- Fabrizio Gilardi, Meysam Alizadeh, and Maël Kubli. Chatgpt outperforms crowd workers for text-annotation tasks. *Proceedings of the National Academy of Sciences*, 120(30):e2305016120, 2023. doi: 10.1073/pnas.2305016120. URL <https://www.pnas.org/doi/abs/10.1073/pnas.2305016120>.
- Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT Press, 2016. <http://www.deeplearningbook.org>.
- Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial networks. *Commun. ACM*, 63(11):139–144, oct 2020. ISSN 0001-0782. doi: 10.1145/3422622. URL <https://doi.org/10.1145/3422622>.
- Johannes Graën, Mara Bertamini, and Martin Volk. Cutter – a universal multilingual tokenizer. In Mark Cieliebak, Don Tuggener, and Fernando Benites, editors, *Swiss Text Analytics Conference*, number 2226 in CEUR Workshop Proceedings, pages 75–81. CEUR-WS, June 2018. URL <https://doi.org/10.5167/uzh-157243>.
- Jiaqi Guan, Runzhe Li, Sheng Yu, and Xuegong Zhang. A method for generating synthetic electronic medical record text. *IEEE/ACM Transactions on Computational Biology and Bioinformatics*, 18(1):173–182, 2021. doi: 10.1109/TCBB.2019.2948985.
- Kai Hakala and Sampo Pyysalo. Biomedical named entity recognition with multilingual BERT. In Kim Jin-Dong, Nédellec Claire, Bossy Robert, and Deléger Louise, editors, *Proceedings of the 5th Workshop on BioNLP Open Shared Tasks*, pages 56–61, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-5709. URL <https://aclanthology.org/D19-5709>.

- Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. {DEBERTA}: {DECODING}-{enhanced} {bert} {with} {disentangled} {attention}. In *International Conference on Learning Representations*, 2021. URL <https://openreview.net/forum?id=XPZiaotutsD>.
- Xingwei He, Zhenghao Lin, Yeyun Gong, A-Long Jin, Hang Zhang, Chen Lin, Jian Jiao, Siu Ming Yiu, Nan Duan, and Weizhu Chen. AnnoLLM: Making large language models to be better crowdsourced annotators. In Yi Yang, Aida Davani, Avi Sil, and Anoop Kumar, editors, *Proceedings of the 2024 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (Volume 6: Industry Track)*, pages 165–190, Mexico City, Mexico, June 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.naacl-industry.15>.
- Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9(8): 1735–1780, nov 1997. ISSN 0899-7667. doi: 10.1162/neco.1997.9.8.1735. URL <https://doi.org/10.1162/neco.1997.9.8.1735>.
- Ari Holtzman, Jan Buys, Li Du, Margaret Forbes, and Yejin Choi. The curious case of neural text degeneration. In *International Conference on Learning Representations*, 2020.
- Matthew Honnibal and Ines Montani. spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing. To appear, 2017.
- Kexin Huang, Jaan Altosaar, and R. Ranganath. Clinicalbert: Modeling clinical notes and predicting hospital readmission. *ArXiv*, abs/1904.05342, 2019.
- International Organization for Standards (ISO). Data elements and interchange formats—Information interchange—Representation of dates and times. Technical report, International Organization for Standards, 2004. ISO 8601.
- Ievgen Iosifov, Olena Iosifova, and Volodymyr Sokolov. Sentence segmentation from unformatted text using language modeling and sequence labeling approaches. In *2020 IEEE International Conference on Problems of Infocommunications. Science and Technology (PIC ST)*, pages 335–337, 2020. doi: 10.1109/PICST51311.2020.9468084.
- Patthamanan Isaranontakul and Worapoj Kreesuradej. A study of using gpt-3 to generate a thai sentiment analysis of covid-19 tweets dataset. In *2023 20th International Joint Conference on Computer Science and Software Engineering (JCSSE)*, pages 106–111, 2023. doi: 10.1109/JCSSE58229.2023.10201994.
- Julia Ive, Natalia Viani, Joyce Kam, Lucia Yin, Somain Verma, Stephen Puntis, Rudolf N Cardinal, Angus Roberts, Robert Stewart, and Sumithra Velupillai. Generation and evaluation of artificial mental health records for natural language processing. *npj Digital Medicine*, 3(1):69, May 2020.
- Abdul Jabbar, Sajid Iqbal, Manzoor Ilahi Tamimy, Shafiq Hussain, and Adnan Akhunzada. Empirical evaluation and study of text stemming algorithms. *Artificial Intelligence Review*, 53(8): 5559–5588, December 2020.
- M. Jayabalan and M. E. Rana. Anonymizing healthcare records: A study of privacy preserving data publishing techniques. *Advances in Science Letters*, 24(3):1694–1697, 2018. doi: 10.1166/asl.2018.11139. URL <http://dx.doi.org/10.1166/asl.2018.11139>.
- F. Jelinek, R. L. Mercer, L. R. Bahl, and J. K. Baker. Perplexity—a measure of the difficulty of speech recognition tasks. *J. Acoust. Soc. Am.*, 62(S1):S63, 12 1977. doi: 10.1121/1.2016299.

- Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Ye Jin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *ACM Comput. Surv.*, 55(12), mar 2023. ISSN 0360-0300. doi: 10.1145/3571730. URL <https://doi.org/10.1145/3571730>.
- Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition)*. Prentice-Hall, Inc., USA, 2009. ISBN 0131873210.
- Nitish Shirish Keskar, Bryan McCann, Lav R. Varshney, Caiming Xiong, and Richard Socher. Ctrl: A conditional transformer language model for controllable generation, 2019. URL <https://arxiv.org/abs/1909.05858>.
- Jin-Dong Kim, Tomoko Ohta, and Jun’ichi Tsujii. Corpus annotation for mining biomedical events from literature. *BMC Bioinformatics*, 9(1):10, January 2008.
- Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *International Conference on Learning Representations (ICLR)*, San Diego, CA, USA, 2015.
- Tibor Kiss and Jan Strunk. Unsupervised multilingual sentence boundary detection. *Computational Linguistics*, 32(4):485–525, 2006. doi: 10.1162/coli.2006.32.4.485. URL <https://aclanthology.org/J06-4003>.
- Jan-Christoph Klie, Michael Bugert, Beto Boullosa, Richard Eckart de Castilho, and Iryna Gurevych. The INCEPTION platform: Machine-assisted and knowledge-oriented interactive annotation. In Dongyan Zhao, editor, *Proceedings of the 27th International Conference on Computational Linguistics: System Demonstrations*, pages 5–9, Santa Fe, New Mexico, August 2018. Association for Computational Linguistics. URL <https://aclanthology.org/C18-2002>.
- Veysel Kocaman and David Talby. Biomedical named entity recognition at scale. In Alberto Del Bimbo, Rita Cucchiara, Stan Sclaroff, Giovanni Maria Farinella, Tao Mei, Marco Bertini, Hugo Jair Escalante, and Roberto Vezzani, editors, *Pattern Recognition. ICPR International Workshops and Challenges*, pages 635–646, Cham, 2021. Springer International Publishing. ISBN 978-3-030-68763-2.
- Takeshi Kojima, Shixiang Shane Gu, Machel Reid, Yutaka Matsuo, and Yusuke Iwasawa. Large language models are zero-shot reasoners, 2023.
- Taku Kudo. Subword regularization: Improving neural network translation models with multiple subword candidates. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 66–75, Melbourne, Australia, July 2018. Association for Computational Linguistics. doi: 10.18653/v1/P18-1007. URL <https://aclanthology.org/P18-1007>.
- S. Kullback and R. A. Leibler. On Information and Sufficiency. *The Annals of Mathematical Statistics*, 22(1):79 – 86, 1951. doi: 10.1214/aoms/1177729694. URL <https://doi.org/10.1214/aoms/1177729694>.
- Dhara J. Ladani and Nikita P. Desai. Stopword identification and removal techniques on tc and ir applications: A survey. In *2020 6th International Conference on Advanced Computing and Communication Systems (ICACCS)*, pages 466–472, 2020. doi: 10.1109/ICACCS48705.2020.9074166.

- John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. Conditional random fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning, ICML '01*, page 282–289, San Francisco, CA, USA, 2001. Morgan Kaufmann Publishers Inc. ISBN 1558607781.
- Jinhyuk Lee, Wonjin Yoon, Sungdong Kim, Donghyeon Kim, Sunkyu Kim, Chan Ho So, and Jaewoo Kang. Biobert: a pre-trained biomedical language representation model for biomedical text mining. *Bioinformatics*, 36:1234 – 1240, 2019. doi: 10.1093/bioinformatics/btz682.
- Scott H Lee. Natural language generation for electronic health records. *npj Digital Medicine*, 1(1):63, November 2018.
- Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks. In *Proceedings of the 34th International Conference on Neural Information Processing Systems, NIPS'20*, Red Hook, NY, USA, 2020. Curran Associates Inc. ISBN 9781713829546.
- Jianfu Li, Yujia Zhou, Xiaoqian Jiang, K. Natarajan, Serguei V. S. Pakhomov, Hongfang Liu, and Hua Xu. Are synthetic clinical notes useful for real natural language processing tasks: A case study on clinical entity recognition. *Journal of the American Medical Informatics Association : JAMIA*, 2021. doi: 10.1093/jamia/ocab112.
- Jin Li, Benjamin J Cairns, Jingsong Li, and Tingting Zhu. Generating synthetic mixed-type longitudinal electronic health records for artificial intelligent applications. *npj Digital Medicine*, 6(1):98, May 2023.
- Claudia Alessandra Libbi, Jan Trienes, Dolf Trieschnigg, and Christin Seifert. Generating synthetic training data for supervised de-identification of electronic health records. *Future Internet*, 13(5), 2021. ISSN 1999-5903. doi: 10.3390/fi13050136. URL <https://www.mdpi.com/1999-5903/13/5/136>.
- Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL <https://aclanthology.org/W04-1013>.
- Linguistic Data Consortium (LDC). *Ace (Automatic Content Extraction) English Annotation Guidelines for Events*, 2005.
- Jiacheng Liu, Alisa Liu, Ximing Lu, Sean Welleck, Peter West, Ronan Le Bras, Yejin Choi, and Hannaneh Hajishirzi. Generated knowledge prompting for commonsense reasoning. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3154–3169, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.225. URL <https://aclanthology.org/2022.acl-long.225>.
- Xiong Liu, Greg L. Hersch, Iya Khalil, and Murthy Devarakonda. Clinical trial information extraction with bert. In *2021 IEEE 9th International Conference on Healthcare Informatics (ICHI)*, pages 505–506, Victoria, BC, Canada, 2021. doi: 10.1109/ICHI52183.2021.00092.

- Fábio Lopes, César Teixeira, and Hugo Gonçalo Oliveira. Contributions to clinical named entity recognition in Portuguese. In Dina Demner-Fushman, Kevin Bretonnel Cohen, Sophia Ananiadou, and Junichi Tsujii, editors, *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 223–233, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5024. URL <https://aclanthology.org/W19-5024>.
- Fábio Lopes, César Teixeira, and Hugo Gonçalo Oliveira. Comparing different methods for named entity recognition in portuguese neurology text. *Journal of Medical Systems*, 44(4):77, February 2020.
- Qiuhaio Lu, Dejing Dou, and Thien Huu Nguyen. Textual data augmentation for patient outcomes prediction. In *2021 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 2817–2821, Houston, TX, USA, 2021. doi: 10.1109/BIBM52615.2021.9669861.
- Yao Lu, Max Bartolo, Alastair Moore, Sebastian Riedel, and Pontus Stenetorp. Fantastically ordered prompts and where to find them: Overcoming few-shot prompt order sensitivity. In Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, editors, *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8086–8098, Dublin, Ireland, May 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.acl-long.556. URL <https://aclanthology.org/2022.acl-long.556>.
- Mehdi Mirza and Simon Osindero. Conditional generative adversarial nets, 2014.
- Anders Giovanni Møller, Arianna Pera, Jacob Dalsgaard, and Luca Aiello. The parrot dilemma: Human-labeled vs. LLM-augmented data in classification tasks. In Yvette Graham and Matthew Purver, editors, *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 179–192, St. Julian’s, Malta, March 2024. Association for Computational Linguistics. URL <https://aclanthology.org/2024.eacl-short.17>.
- Nasrin Mostafazadeh, Nathanael Chambers, Xiaodong He, Devi Parikh, Dhruv Batra, Lucy Vanderwende, Pushmeet Kohli, and James Allen. A corpus and cloze evaluation for deeper understanding of commonsense stories. In Kevin Knight, Ani Nenkova, and Owen Rambow, editors, *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 839–849, San Diego, California, June 2016. Association for Computational Linguistics. doi: 10.18653/v1/N16-1098. URL <https://aclanthology.org/N16-1098>.
- Hajra Murtaza, Musharif Ahmed, Naurin Farooq Khan, Ghulam Murtaza, Saad Zafar, and Ambreen Bano. Synthetic data generation: State of the art in health care domain. *Computer Science Review*, 48:100546, 2023. ISSN 1574-0137. doi: <https://doi.org/10.1016/j.cosrev.2023.100546>. URL <https://www.sciencedirect.com/science/article/pii/S1574013723000138>.
- Mark Neumann, Daniel King, Iz Beltagy, and Waleed Ammar. ScispaCy: Fast and robust models for biomedical natural language processing. In Dina Demner-Fushman, Kevin Bretonnel Cohen, Sophia Ananiadou, and Junichi Tsujii, editors, *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 319–327, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5034. URL <https://aclanthology.org/W19-5034>.

- Lucas Emanuel Silva e Oliveira, Ana Carolina Peters, Adalniza Moura Pucca da Silva, Caroline Pilatti Gebelucá, Yohan Bonescki Gumiel, Lilian Mie Mukai Cintho, Deborah Ribeiro Carvalho, Sadid Al Hasan, and Claudia Maria Cabral Moro. SemClinBr - a multi-institutional and multi-specialty semantically annotated corpus for portuguese clinical NLP tasks. *Journal of Biomedical Semantics*, 13(1):13, May 2022.
- OpenAI, :, Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altschmidt, Sam Altman, Shyamal Anadkat, Red Avila, Igor Babuschkin, Suchir Balaji, Valerie Balcom, Paul Baltescu, Haiming Bao, Mo Bavarian, Jeff Belgum, Irwan Bello, Jake Berdine, Gabriel Bernadett-Shapiro, Christopher Berner, Lenny Bogdonoff, Oleg Boiko, Madelaine Boyd, Anna-Luisa Brakman, Greg Brockman, Tim Brooks, Miles Brundage, Kevin Button, Trevor Cai, Rosie Campbell, Andrew Cann, Brittany Carey, Chelsea Carlson, Rory Carmichael, Brooke Chan, Che Chang, Fotis Chantzis, Derek Chen, Sully Chen, Ruby Chen, Jason Chen, Mark Chen, Ben Chess, Chester Cho, Casey Chu, Hyung Won Chung, Dave Cummings, Jeremiah Currier, Yunxing Dai, Cory Decareaux, Thomas Degry, Noah Deutsch, Damien Deville, Arka Dhar, David Dohan, Steve Dowling, Sheila Dunning, Adrien Ecoffet, Atty Eleti, Tyna Eloundou, David Farhi, Liam Fedus, Niko Felix, Simón Posada Fishman, Juston Forte, Isabella Fulford, Leo Gao, Elie Georges, Christian Gibson, Vik Goel, Tarun Gogineni, Gabriel Goh, Rapha Gontijo-Lopes, Jonathan Gordon, Morgan Grafstein, Scott Gray, Ryan Greene, Joshua Gross, Shixiang Shane Gu, Yufei Guo, Chris Hallacy, Jesse Han, Jeff Harris, Yuchen He, Mike Heaton, Johannes Heidecke, Chris Hesse, Alan Hickey, Wade Hickey, Peter Hoeschele, Brandon Houghton, Kenny Hsu, Shengli Hu, Xin Hu, Joost Huizinga, Shantanu Jain, Shawn Jain, Joanne Jang, Angela Jiang, Roger Jiang, Haozhun Jin, Denny Jin, Shino Jomoto, Billie Jonn, Heewoo Jun, Tomer Kaftan, Łukasz Kaiser, Ali Kamali, Ingmar Kanitscheider, Nitish Shirish Keskar, Tabarak Khan, Logan Kilpatrick, Jong Wook Kim, Christina Kim, Yongjik Kim, Hendrik Kirchner, Jamie Kiros, Matt Knight, Daniel Kokotajlo, Łukasz Kondraciuk, Andrew Kondrich, Aris Konstantinidis, Kyle Kosic, Gretchen Krueger, Vishal Kuo, Michael Lampe, Ikai Lan, Teddy Lee, Jan Leike, Jade Leung, Daniel Levy, Chak Ming Li, Rachel Lim, Molly Lin, Stephanie Lin, Mateusz Litwin, Theresa Lopez, Ryan Lowe, Patricia Lue, Anna Makanju, Kim Malfacini, Sam Manning, Todor Markov, Yaniv Markovski, Bianca Martin, Katie Mayer, Andrew Mayne, Bob McGrew, Scott Mayer McKinney, Christine McLeavey, Paul McMillan, Jake McNeil, David Medina, Aalok Mehta, Jacob Menick, Luke Metz, Andrey Mishchenko, Pamela Mishkin, Vinnie Monaco, Evan Morikawa, Daniel Mossing, Tong Mu, Mira Murati, Oleg Murk, David Mély, Ashvin Nair, Reiichiro Nakano, Rajeev Nayak, Arvind Neelakantan, Richard Ngo, Hyeonwoo Noh, Long Ouyang, Cullen O’Keefe, Jakub Pachocki, Alex Paino, Joe Palermo, Ashley Pantuliano, Giambattista Parascandolo, Joel Parish, Emy Parparita, Alex Passos, Mikhail Pavlov, Andrew Peng, Adam Perelman, Filipe de Avila Belbute Peres, Michael Petrov, Henrique Ponde de Oliveira Pinto, Michael, Pokorny, Michelle Pokrass, Vitchyr Pong, Tolly Powell, Alethea Power, Boris Power, Elizabeth Proehl, Raul Puri, Alec Radford, Jack Rae, Aditya Ramesh, Cameron Raymond, Francis Real, Kendra Rimbach, Carl Ross, Bob Rotsted, Henri Roussez, Nick Ryder, Mario Saltarelli, Ted Sanders, Shibani Santurkar, Girish Sastry, Heather Schmidt, David Schnurr, John Schulman, Daniel Selsam, Kyla Sheppard, Toki Sherbakov, Jessica Shieh, Sarah Shoker, Pranav Shyam, Szymon Sidor, Eric Sigler, Maddie Simens, Jordan Sitkin, Katarina Slama, Ian Sohl, Benjamin Sokolowsky, Yang Song, Natalie Staudacher, Felipe Petroski Such, Natalie Summers, Ilya Sutskever, Jie Tang, Nikolas Tezak, Madeleine Thompson, Phil Tillet, Amin Tootoonchian, Elizabeth Tseng, Preston Tuggle, Nick Turley, Jerry Tworek, Juan Felipe Cerón Uribe, Andrea Vallone, Arun Vijayvergiya, Chelsea Voss, Carroll Wainwright, Justin Jay Wang, Alvin Wang, Ben Wang, Jonathan Ward, Jason Wei, CJ Weinmann, Akila

- Welihinda, Peter Welinder, Jiayi Weng, Lilian Weng, Matt Wiethoff, Dave Willner, Clemens Winter, Samuel Wolrich, Hannah Wong, Lauren Workman, Sherwin Wu, Jeff Wu, Michael Wu, Kai Xiao, Tao Xu, Sarah Yoo, Kevin Yu, Qiming Yuan, Wojciech Zaremba, Rowan Zellers, Chong Zhang, Marvin Zhang, Shengjia Zhao, Tianhao Zheng, Juntang Zhuang, William Zhuk, and Barret Zoph. Gpt-4 technical report, 2023.
- Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. Training language models to follow instructions with human feedback. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc., 2022. URL [https://proceedings.neurips.cc/paper\\_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2022/file/b1efde53be364a73914f58805a001731-Paper-Conference.pdf).
- David D. Palmer. Chapter 2 : Tokenisation and sentence segmentation. 2007. URL <https://api.semanticscholar.org/CorpusID:37409158>.
- Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, ACL '02, page 311–318, USA, 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL <https://doi.org/10.3115/1073083.1073135>.
- Cheng Peng, Xi Yang, Aokun Chen, Kaleb E Smith, Nima PourNejatian, Anthony B Costa, Cheryl Martin, Mona G Flores, Ying Zhang, Tanja Magoc, Gloria Lipori, Duane A Mitchell, Naykky S Ospina, Mustafa M Ahmed, William R Hogan, Elizabeth A Shenkman, Yi Guo, Jiang Bian, and Yonghui Wu. A study of generative large language model for medical research and healthcare. *npj Digital Medicine*, 6(1):210, November 2023.
- Yifan Peng, Shankai Yan, and Zhiyong Lu. Transfer learning in biomedical natural language processing: An evaluation of BERT and ELMo on ten benchmarking datasets. In Dina Demner-Fushman, Kevin Bretonnel Cohen, Sophia Ananiadou, and Junichi Tsujii, editors, *Proceedings of the 18th BioNLP Workshop and Shared Task*, pages 58–65, Florence, Italy, August 2019. Association for Computational Linguistics. doi: 10.18653/v1/W19-5006. URL <https://aclanthology.org/W19-5006>.
- Steven T Piantadosi. Zipf’s word frequency law in natural language: a critical review and future directions. *Psychon. Bull. Rev.*, 21(5):1112–1130, October 2014.
- M F Porter. An algorithm for suffix stripping. *Program*, 14(3):130–137, January 1980.
- James Pustejovsky, José M. Castaño, Robert Ingria, Roser Sauri, Robert J. Gaizauskas, Andrea Setzer, Graham Katz, and Dragomir R. Radev. TimeML: Robust specification of event and temporal expressions in text. In *New Directions in Question Answering*, pages 28–34, 2003. 2003a.
- James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Saurí. Temporal and event information in natural language text. *Language Resources and Evaluation*, 39(2):123–164, May 2005.

- Pedro Vitor Quinta de Castro, Nádia Félix Felipe da Silva, and Anderson da Silva Soares. Portuguese named entity recognition using lstm-crf. In Aline Villavicencio, Viviane Moreira, Alberto Abad, Helena Caseli, Pablo Gamallo, Carlos Ramisch, Hugo Gonçalo Oliveira, and Gustavo Henrique Paetzold, editors, *Computational Processing of the Portuguese Language*, pages 83–92, Cham, 2018. Springer International Publishing. ISBN 978-3-319-99722-3.
- Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. 2018.
- Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.
- John J Riva, Keshena M P Malik, Stephen J Burnie, Andrea R Endicott, and Jason W Busse. What is your research question? an introduction to the PICOT format for clinicians. *J Can Chiropr Assoc*, 56(3):167–171, September 2012.
- João Rodrigues, Luís Gomes, João Silva, António Branco, Rodrigo Santos, Henrique Lopes Cardoso, and Tomás Osório. *Advancing Neural Encoding of Portuguese with Transformer Albertina PT-\**, page 441–453. Springer Nature Switzerland, 2023. ISBN 9783031490088. doi: 10.1007/978-3-031-49008-8\_35. URL [http://dx.doi.org/10.1007/978-3-031-49008-8\\_35](http://dx.doi.org/10.1007/978-3-031-49008-8_35).
- Ricardo Rodrigues, Hugo Gonçalo Oliveira, and Paulo Gomes. NLPPort: A Pipeline for Portuguese NLP. In Pedro Rangel Henriques, José Paulo Leal, António Menezes Leitão, and Xavier Gómez Guinovart, editors, *7th Symposium on Languages, Applications and Technologies (SLATE 2018)*, volume 62 of *Open Access Series in Informatics (OASICS)*, pages 18:1–18:9, Dagstuhl, Germany, 2018. Schloss Dagstuhl – Leibniz-Zentrum für Informatik. ISBN 978-3-95977-072-9. doi: 10.4230/OASICS.SLATE.2018.18. URL <https://drops.dagstuhl.de/entities/document/10.4230/OASICS.SLATE.2018.18>.
- Wei Ruan and Won-sook Lee. Recognising named entity of medical imaging procedures in clinical notes. In *2018 IEEE International Conference on Bioinformatics and Biomedicine (BIBM)*, pages 743–746, Madrid, Spain, 2018. doi: 10.1109/BIBM.2018.8621452.
- Sebastian Ruder. An overview of gradient descent optimization algorithms. *CoRR*, abs/1609.04747, 2016. URL <http://arxiv.org/abs/1609.04747>.
- Brenda Santana, Ricardo Campos, Evelin Amorim, Alípio Jorge, Purificação Silvano, and Sérgio Nunes. A survey on narrative extraction from textual data. *Artificial Intelligence Review*, 56(8): 8393–8435, August 2023.
- Joaquim Santos, Bernardo Consoli, Cicero dos Santos, Juliano Terra, Sandra Collonini, and Renata Vieira. Assessing the impact of contextual embeddings for portuguese named entity recognition. In *2019 8th Brazilian Conference on Intelligent Systems (BRACIS)*, pages 437–442, 2019. doi: 10.1109/BRACIS.2019.00083.
- Timo Schick and Hinrich Schütze. Generating datasets with pretrained language models. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6943–6951, Online and Punta Cana, Dominican Republic, November 2021a. Association for Computational Linguistics. doi: 10.18653/v1/2021.emnlp-main.555. URL <https://aclanthology.org/2021.emnlp-main.555>.

- Timo Schick and Hinrich Schütze. Exploiting cloze-questions for few-shot text classification and natural language inference. In Paola Merlo, Jorg Tiedemann, and Reut Tsarfaty, editors, *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 255–269, Online, April 2021b. Association for Computational Linguistics. doi: 10.18653/v1/2021.eacl-main.20. URL <https://aclanthology.org/2021.eacl-main.20>.
- Elisa Terumi Rubel Schneider, João Vitor Andrioli de Souza, Julien Knafo, Lucas Emanuel Silva e Oliveira, Jenny Copara, Yohan Bonescki Gumiel, Lucas Ferro Antunes de Oliveira, Emerson Cabrera Paraiso, Douglas Teodoro, and Cláudia Maria Cabral Moro Barra. BioBERTpt - a Portuguese neural language model for clinical named entity recognition. In Anna Rumshisky, Kirk Roberts, Steven Bethard, and Tristan Naumann, editors, *Proceedings of the 3rd Clinical Natural Language Processing Workshop*, pages 65–72, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.clinicalnlp-1.7. URL <https://aclanthology.org/2020.clinicalnlp-1.7>.
- John Schulman, Filip Wolski, Prafulla Dhariwal, Alec Radford, and Oleg Klimov. Proximal policy optimization algorithms, 2017.
- Isabel Segura-Bedmar, Paloma Martínez, and María Herrero-Zazo. SemEval-2013 task 9 : Extraction of drug-drug interactions from biomedical texts (DDIExtraction 2013). In Suresh Manandhar and Deniz Yuret, editors, *Second Joint Conference on Lexical and Computational Semantics (\*SEM), Volume 2: Proceedings of the Seventh International Workshop on Semantic Evaluation (SemEval 2013)*, pages 341–350, Atlanta, Georgia, USA, June 2013. Association for Computational Linguistics. URL <https://aclanthology.org/S13-2056>.
- Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In Katrin Erk and Noah A. Smith, editors, *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1715–1725, Berlin, Germany, August 2016. Association for Computational Linguistics. doi: 10.18653/v1/P16-1162. URL <https://aclanthology.org/P16-1162>.
- Jianlin Shi, Danielle Mowery, Katherine M. Doing-Harris, and John F. Hurdle. Rush: a rule-based segmentation tool using hashing for extremely accurate sentence segmentation of clinical text. In *AMIA Annual Symposium Proceedings*, 2016.
- Matthew Snover, Bonnie Dorr, Rich Schwartz, Linnea Micciulla, and John Makhoul. A study of translation edit rate with targeted human annotation. In *Proceedings of the 7th Conference of the Association for Machine Translation in the Americas: Technical Papers*, pages 223–231, Cambridge, Massachusetts, USA, August 8-12 2006. Association for Machine Translation in the Americas. URL <https://aclanthology.org/2006.amta-papers.25>.
- Oswaldo Solarte-Pabón, Alberto Blazquez-Herranz, Maria Torrente, Alejandro Rodríguez-Gonzalez, Mariano Provencio, and Ernestina Menasalvas. Extracting cancer treatments from clinical text written in spanish: A deep learning approach. In *2021 IEEE 8th International Conference on Data Science and Advanced Analytics (DSAA)*, pages 1–6, Porto, Portugal, 2021. doi: 10.1109/DSAA53316.2021.9564137.
- Fábio Souza, Rodrigo Frassetto Nogueira, and Roberto de Alencar Lotufo. Portuguese named entity recognition using BERT-CRF. *CoRR*, abs/1909.10649, 2019. URL <http://arxiv.org/abs/1909.10649>.

- Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014. URL <http://jmlr.org/papers/v15/srivastava14a.html>.
- Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Language Resources and Evaluation*, 47(2):269–298, June 2013.
- Ruixiang Tang, Xiaotian Han, Xiaoqian Jiang, and Xia Hu. Does synthetic data generation of llms help clinical text mining? *ArXiv*, abs/2303.04360, 2023. doi: 10.48550/arXiv.2303.04360.
- Wilson L. Taylor. “cloze procedure”: A new tool for measuring readability. *Journalism Quarterly*, 30(4):415–433, 1953. doi: 10.1177/107769905303000401. URL <https://doi.org/10.1177/107769905303000401>.
- Erik F. Tjong Kim Sang. Introduction to the CoNLL-2002 shared task: Language-independent named entity recognition. In *COLING-02: The 6th Conference on Natural Language Learning 2002 (CoNLL-2002)*, 2002. URL <https://aclanthology.org/W02-2024>.
- Erik F. Tjong Kim Sang and Sabine Buchholz. Introduction to the CoNLL-2000 shared task chunking. In *Fourth Conference on Computational Natural Language Learning and the Second Learning Language in Logic Workshop*, 2000. URL <https://aclanthology.org/W00-0726>.
- Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003. URL <https://aclanthology.org/W03-0419>.
- Yitong Tseo, M. I. Salkola, Ahmed Mohamed, Anuj Kumar, and Freddy Abnoui. Information extraction of clinical trial eligibility criteria, 2020. URL <https://arxiv.org/abs/2006.07296>.
- Hangyu Tu, Lifeng Han, and Goran Nenadic. Extraction of medication and temporal relation from clinical text using neural language models. In *2023 IEEE International Conference on Big Data (BigData)*, pages 2735–2744, Sorrento, Italy, 2023. doi: 10.1109/BigData59044.2023.10386489.
- S. Ubani, S. Polat, and Rodney D. Nielsen. Zeroshotdataaug: Generating and augmenting training data with chatgpt. *ArXiv*, abs/2304.14334, 2023. doi: 10.48550/arXiv.2304.14334.
- M. Saef Ullah Miah, Junaida Sulaiman, Talha Bin Sarwar, Saima Sharleen Islam, Mizanur Rahman, and Md. Samiul Haque. Medical named entity recognition (medner): A deep learning model for recognizing medical entities (drug, disease) from scientific texts. In *IEEE EUROCON 2023 - 20th International Conference on Smart Technologies*, pages 158–162, Torino, Italy, 2023. doi: 10.1109/EUROCON56442.2023.10199075.
- Özlem Uzuner, Brett R South, Shuying Shen, and Scott L DuVall. 2010 i2b2/VA challenge on concepts, assertions, and relations in clinical text. *J Am Med Inform Assoc*, 18(5):552–556, June 2011.

- Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*, NIPS'17, page 6000–6010, Red Hook, NY, USA, 2017. Curran Associates Inc. ISBN 9781510860964.
- Marc Verhagen, Robert Gaizauskas, Frank Schilder, Mark Hepple, Jessica Moszkowicz, and James Pustejovsky. The TempEval challenge: identifying temporal relations in text. *Language Resources and Evaluation*, 43(2):161–179, June 2009.
- Jason Walonoski, Mark Kramer, Joseph Nichols, Andre Quina, Chris Moesel, Dylan Hall, Carlton Duffett, Kudakwashe Dube, Thomas Gallagher, and Scott McLachlan. Synthea: An approach, method, and software mechanism for generating synthetic patients and the synthetic electronic health care record. *Journal of the American Medical Informatics Association*, 25(3):230–238, 08 2017. ISSN 1527-974X. doi: 10.1093/jamia/ocx079. URL <https://doi.org/10.1093/jamia/ocx079>.
- Xia Wang, Jia Wei, Dale Schuurmans, Quoc V Le, Ed H Chi, Shashi Narang, et al. Self-consistency improves chain of thought reasoning in language models. In *Eleventh International Conference on Learning Representations*, 2023.
- Albert Webson and Ellie Pavlick. Do prompt-based models really understand the meaning of their prompts? In Marine Carpuat, Marie-Catherine de Marneffe, and Ivan Vladimir Meza Ruiz, editors, *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 2300–2344, Seattle, United States, July 2022. Association for Computational Linguistics. doi: 10.18653/v1/2022.naacl-main.167. URL <https://aclanthology.org/2022.naacl-main.167>.
- Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, brian ichter, Fei Xia, Ed Chi, Quoc V Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. In S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh, editors, *Advances in Neural Information Processing Systems*, volume 35, pages 24824–24837. Curran Associates, Inc., 2022. URL [https://proceedings.neurips.cc/paper\\_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf](https://proceedings.neurips.cc/paper_files/paper/2022/file/9d5609613524ecf4f15af0f7b31abca4-Paper-Conference.pdf).
- Rachel Wicks and Matt Post. A unified approach to sentence segmentation of punctuated text in many languages. In Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, editors, *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 3995–4007, Online, August 2021. Association for Computational Linguistics. doi: 10.18653/v1/2021.acl-long.309. URL <https://aclanthology.org/2021.acl-long.309>.
- Dingjun Wu, Jing Zhang, and Xinmei Huang. Chain of thought prompting elicits knowledge augmentation. In Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, editors, *Findings of the Association for Computational Linguistics: ACL 2023*, pages 6519–6534, Toronto, Canada, July 2023. Association for Computational Linguistics. doi: 10.18653/v1/2023.findings-acl.408. URL <https://aclanthology.org/2023.findings-acl.408>.
- Andrew Yale, Saloni Dash, Ritik Dutta, Isabelle Guyon, Adrien Pavao, and Kristin P. Bennett. Generation and evaluation of privacy preserving synthetic health data. *Neurocomputing*, 416:244–255, 2020. ISSN 0925-2312. doi: <https://doi.org/10.1016/j.neucom>.

- 2019.12.136. URL <https://www.sciencedirect.com/science/article/pii/S0925231220305117>.
- Jinsung Yoon, Michel Mizrahi, Nahid Farhady Ghalaty, Thomas Jarvinen, Ashwin S Ravi, Peter Brune, Fanyu Kong, Dave Anderson, George Lee, Arie Meir, Farhana Bandukwala, Elli Kanal, Sercan Ö Arik, and Tomas Pfister. EHR-Safe: generating high-fidelity and privacy-preserving synthetic electronic health records. *npj Digital Medicine*, 6(1):141, August 2023.
- Qiang Zhang, Yong Sun, Linlin Zhang, Yanfei Jiao, and Yue Tian. Named entity recognition method in health preserving field based on bert. *Procedia Computer Science*, 183:212–220, 2021. ISSN 1877-0509. doi: <https://doi.org/10.1016/j.procs.2021.03.010>. URL <https://www.sciencedirect.com/science/article/pii/S1877050921006189>. Proceedings of the 10th International Conference of Information and Communication Technology.