

Identifying References to Legal Literature in Portuguese Superior Court Decisions

Fábio Miguel Pereira Nogueira

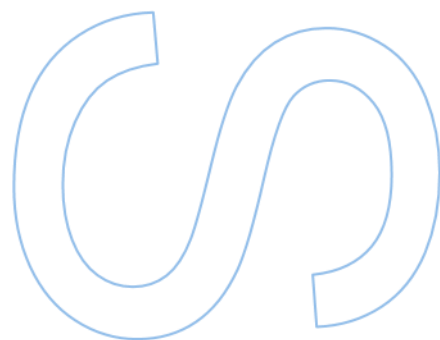
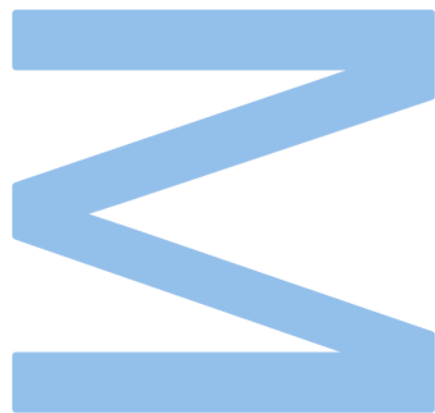
Master's Degree in Network and
Information Systems Engineering
Computer Science Department
2023

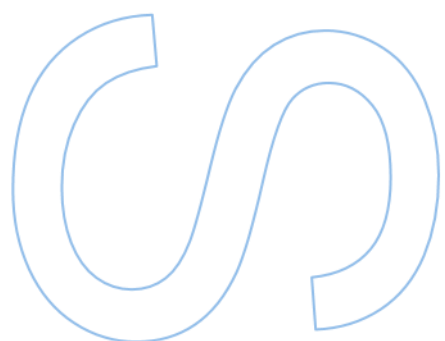
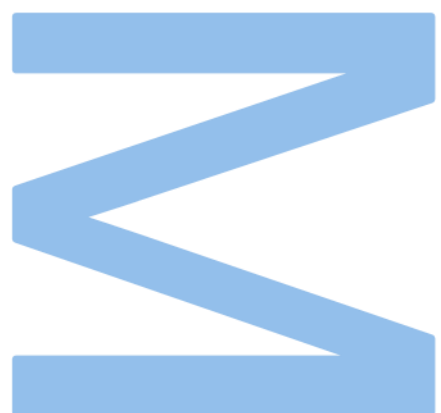
Supervisor

Fernando Silva, Full Professor,
Faculty of Sciences of the University of Porto

Co-supervisor

Maria Raquel Guimarães, Associate Professor
Faculty of Law of the University of Porto / CIJ





Aos meus queridos pais, David e Rosa, cujo apoio inabalável e esforços incansáveis tornaram possível que eu pudesse escrever estas palavras. A vossa orientação e sacrifícios altruístas têm sido inestimáveis. Serão sempre um pilar no meu sucesso.

Obrigado por tudo. Amo-vos.

Sworn Statement

I, Fábio Miguel Pereira Nogueira, enrolled in the Master Degree in Network and Information Systems Engineering at the Faculty of Sciences of the University of Porto hereby declare, in accordance with the provisions of paragraph a) of Article 14 of the Code of Ethical Conduct of the University of Porto, that the content of this dissertation report reflects perspectives, research work and my own interpretations at the time of its submission.

By submitting this dissertation report, I also declare that it contains the results of my own research work and contributions that have not been previously submitted to this or any other institution.

I further declare that all references to other authors fully comply with the rules of attribution and are referenced in the text by citation and identified in the bibliographic references section. This dissertation report does not include any content whose reproduction is protected by copyright laws. I am aware that the practice of plagiarism and self-plagiarism constitute a form of academic offense.

Fábio Miguel Pereira Nogueira

30th of June, 2023

Acknowledgements

First and foremost I would like to express my heartfelt appreciation to Professor Fernando Silva. This thesis has benefited greatly from his exceptional coordination, meticulous proofreading, and unwavering support. I am deeply thankful for his guidance and belief in my abilities as a working student striving to complete this thesis within this year.

Furthermore, I would like to extend my appreciation to Professor Maria Raquel Guimarães. Her expertise in complex legal verbiage and providing essential information on the field was instrumental in shaping this work. I also want to specially thank Sylwia Bugla, from INESC TEC, for her tireless efforts in sentence validation, beyond her support. A special mention goes to Engineer Augusto Ribeiro for his help gathering and providing the authors' list from the FDUP's platform.

I would also like to take this opportunity to express my profound gratitude to my closest family. Their unwavering support, encouragement, and love have been the driving force behind my accomplishments. I am eternally grateful to my my parents for everything, to my brother Rafael for being not only a sibling but also a friend, and to my girlfriend Joana for her unwavering support and understanding throughout this challenging journey.

I am also indebted to all of my friends who have played an integral role in my (not only) academic life. I would like to extend my sincere thanks to Francisco, André, Rui, Rúben, and countless others who have provided support, companionship, and encouragement. Each one of them holds a special place in my heart, and their friendship has enriched my life immeasurably.

Lastly, I would like to acknowledge every single person who has believed in me. Their faith in my abilities have been a constant source of motivation. I would also like to express my gratitude to those who may have doubted me, as their skepticism has pushed me to prove myself and reach new heights. I firmly believe that it is through the challenges and obstacles we face that we grow and evolve, and for that, I am immensely grateful.

Abstract

The decision-making process by judges in the court of law is often influenced by academic legal writings. Hence, their decisions may include references to legal books, research articles and other academic writings. The creation of a citation index based on the jurisprudence of higher courts, possibly organized according to the thematic areas of research, makes it possible to evaluate the impact of the scientific production of jurists who dedicate their careers to the study of Law, namely professors, researchers or graduate students.

The automatic identification of citations in texts is a complex problem. Not only is it necessary to delimit the reference made to legal academic writings in the text, but it is also necessary to identify the authorship by associating it with one or more known authors. This leads to the problem of disambiguating names. The references produced in a text do not always follow the same pattern, and there may be multiple ways of referring to the same author. When abbreviated forms of names are used, the same abbreviation may correspond to two or more different authors, making it difficult to credit the citation of a publication to its true author.

This dissertation tackles the problem of automatically identifying citations, or references, to academic legal writings in superior court decisions and crediting them to their authors. Our proposal is to solve this problem by creating a digital infrastructure using non-relational databases to manage reference information, creating algorithms to identify and clean up possible references within a document, and associating them with authors or researchers.

Our contributions include the development of an algorithm based on author names and citation patterns in legal contexts, an algorithm based on grammar rules from NLP's Named-Entity Recognition that uses chunks created from legal context, and finally a learning algorithm based on a transformer architecture, capable of identifying citations to legal documents within pre-processed sentences extracted from supreme court decisions. We also present an Information Extraction technique based on grammar rules to fetch metadata on a given citation. Performance results are provided for all citation identification strategies.

Keywords: Text Mining, Citation Identification, Citation Extraction, Data Mining, NoSQL, Natural Language Processing, Transformer Architecture.

Resumo

É comum um tribunal fundamentar a sua decisão em citações de trabalhos de investigação produzida por juristas. A criação de um índice de citações com base na jurisprudência dos tribunais superiores, eventualmente organizado em função das áreas temáticas de investigação, possibilita a avaliação do impacto da produção científica dos juristas que dedicam a sua carreira ao estudo do Direito, nomeadamente docentes, investigadores ou estudantes de pós-graduação.

A identificação automática de citações em textos é um problema complexo. Não só é necessário delimitar no texto a referência que é feita a outros trabalhos, como é necessário associar a um ou mais autores deparando-nos frequentemente com o problema de desambiguar nomes. As referências produzidas num texto nem sempre seguem o mesmo padrão, podendo haver múltiplas formas de referir um mesmo autor. Quando são usadas formas abreviadas de nomes, pode dar-se a situação de uma mesma abreviação poder corresponder a dois ou mais autores distintos, dificultando a tarefa de creditação da citação de uma publicação ao seu verdadeiro autor.

Esta dissertação aborda o problema da identificação automática de citações, ou referências, a textos jurídicos académicos em decisões de tribunais superiores e da atribuição de créditos aos seus autores. A nossa proposta é resolver este problema através da criação de uma infraestrutura digital usando bases de dados não relacionais para gerir a informação de referência, criando algoritmos para identificar e limpar possíveis referências dentro de um documento, e associando-as a autores ou investigadores.

Os nossos contributos incluem o desenvolvimento de um algoritmo baseado em nomes de autores e padrões de citação em contextos jurídicos, um algoritmo baseado em regras gramaticais do Reconhecimento de Entidades Nomeadas da PNL que utiliza chunks criados a partir do contexto jurídico e, finalmente, um algoritmo de aprendizagem baseado numa arquitetura transformadora, capaz de identificar citações a documentos jurídicos em frases pré-processadas extraídas de decisões do Supremo Tribunal. Apresentamos também uma técnica de extração de informação baseada em regras gramaticais para obter metadados sobre uma dada citação. São apresentados resultados de desempenho para todas as estratégias de identificação de citações.

Palavras-Chave: Text Mining, Citation Identification, Citation Extraction, Data Mining, NoSQL, Natural Language Processing, Transformer Architecture.

Contents

Sworn Statement	iii
Acknowledgements	v
Abstract	vii
Resumo	ix
Contents	xiv
List of Tables	xv
List of Figures	xviii
Listings	xix
Acronyms	xxi
1 Introduction	1
1.1 Context and Motivation	1
1.2 Main goals	2
1.3 Contributions	4
1.4 Dissertation structure	4
2 Background and Related Work	7
2.1 Data and Text Mining	7

2.2	Data Mining Pipeline	8
2.3	Data Preprocessing and Preparation	10
2.4	Non-relational Databases - NoSQL	12
2.4.1	Types of Non-relational Databases	12
2.4.2	Different cases of NoSQL databases	12
2.5	Natural Language Processing and Large Language Models	13
2.5.1	NLP Techniques	14
2.5.2	NLP Technologies	15
2.6	Transformer architecture	16
2.6.1	BERT - Bidirectional Encoder Representations from Transformers	17
2.6.2	GPT - A State-of-the-Art Language Model	19
2.6.3	Other Models and Applications	20
2.7	Related Work	20
2.7.1	Citeseer - An Automatic Citation Indexing System	20
2.7.2	CiteSeerX	21
2.8	Summary	22
3	Data Preparation and Preprocessing	23
3.1	Data Gathering - Web Scraper	23
3.1.1	Technologies Background	24
3.1.2	Extraction	25
3.2	Data preparation	25
3.3	Database Architecture	26
3.4	Data Preprocessing	31
3.4.1	Data Preparation and Tokenization	31
3.4.2	Data Preparation to Sentence Divider	32
3.4.3	Authors' Data	32
3.5	Summary	34

4	Identification of references to legal literature	35
4.1	Sentence Tokenization	36
4.2	Data preparation for extraction	38
4.2.1	Output of identified references	39
4.2.2	BERT	40
4.3	Summary	43
5	References Extraction	45
5.1	Extraction Tool Preparation	45
5.2	Grammar structure for chunk identification	47
5.3	Citation information extraction	50
5.4	Summary	51
6	Results	53
6.1	Citation Identification Results Analysis	53
6.1.1	Identification based on patterns and researcher's names	54
6.1.2	Identify by grammar rules	56
6.1.3	BERT	58
6.1.4	Conclusions	60
6.2	Presenting extracted data	61
6.3	Summary	62
7	Conclusion and Future work	63
7.1	Future Work	63
A	Auxiliary Tables	65
B	Auxiliary Graphs	77
C	Auxiliary Code Samples	79

D Auxiliary Images	81
References	83

List of Tables

3.1	Renaming fields from raw dataset to new dataset	26
3.2	Database Architecture - Publication	27
3.3	Database Architecture - Reference	28
3.4	Database Architecture - Author	29
3.5	Database Architecture - Publication-Author	29
4.1	Example text and its transformation into tokens and attention mask	40
A.1	HTML Requests Header for Web Scrapping Algorithm	65
A.2	Example of patterns in european portuguese legal literature for citations	66
A.3	Mac-Morpho's lexical tags and its grammar class	67
A.4	Example of paires "Word-Tag" used in the custom Mac-Morpho list	68
A.5	Citation identification from authors' names and patterns (Raw)	69
A.6	Citation identification from authors' names and patterns (All names)	70
A.7	Citation identification from authors' names and patterns (All names and less Patterns)	71
A.8	Citation identification from authors' names (All names), and Grammar Rules . .	72
A.9	Citation identification from Grammar Rule's Patterns	73
A.10	Citation identification from BERT, iteration 1	74
A.11	Citation identification from BERT, iteration 2	75
A.12	Citation identification from BERT, iteration 3	76

List of Figures

1.1	Project's Pipeline	4
1.2	Dissertation's Pipeline	5
2.1	Cross Industry Standard Process for Data Mining (CRISP-DM)	9
2.2	A 2-D customer data plot with respect to customer locations in a city, showing three data clusters. Outliers may be detected as values that fall outside of the cluster sets.	11
2.3	Comparison of BERT, OpenAI GPT and ELMo	17
2.4	Model structure of the label-masked language model. "N-MASK" is a mask token containing negative label information.	18
3.1	Project's Pipeline: Data Preparation Stage	23
3.2	DGSI's URL Structure	25
3.3	Database schema involving all tables and relations	30
3.4	Architecture of the Punkt System	33
4.1	Project's Pipeline: Citation Identification Stage	35
4.2	Graph plot relating identification via patterns first and names first.	37
4.3	Tokenization process pipeline visualizer	41
4.4	Token IDs' array as a tensor object and their respective class	41
4.5	Validation calculations to process training's accuracy, precision, recall and specificity.	42
5.1	Project's Pipeline: Metadata Extraction Stage	45
5.2	Segmentation and Labeling at both the Token and Chunk Levels	48

5.3	Citation Rule architecture	49
5.4	Citation’s Chunk Tree Example from Python’s draw tool	50
6.1	Project’s Pipeline: Information Presenting Stage	53
6.2	Comparison of validators metrics between different approaches on citation identification	57
6.3	Boxplot on sentence length distribution for all 7434 lines gathered	58
6.4	Comparison of validator’s metrics between different iterations of BERT’s dataset	59
6.5	Comparison of computational times between different iterations of BERT’s dataset	59
6.6	Comparison of validator’s metrics between different epochs’ values on the same iteration of BERT’s dataset	60
6.7	Comparison of times to identify citations in 20 documents between different approaches to the citation identification problem	61
B.1	Graph plot relating identification via patterns first and names first.	77
B.2	Boxplot on sentence length distribution for all 121 lines with citation gathered	78
D.1	DGSI’s URL Structure	81

Listings

C.1 BERT Training	79
C.2 BERT Validation	79
C.3 BERT Usage	80

Acronyms

BERT	Bidirectional Encoder Representations from Transformers	NLP	Natural Language Processing
CoNLL	Conference on Computational Natural Language Learning	NoSQL	Not only SQL
CNN	Convolutional Neural Networks	RNN	Recurrent Neural Network
DM	Data Mining	SQuAD	Stanford Question Answering Dataset
GLUE	General Language Understanding Evaluation	NLTK	Natural Language Toolkit
GPT	Generative Pretrained Transformer	ML	Machine Learning
TM	Text Mining	PoS	Part-of-Speech
LaMDA	Language Model for Dialogue Applications	TP	True Positive
LLM	Large Language Model	FP	False Positive
		TN	True Negative
		FN	False Negative

Chapter 1

Introduction

The automatic identification of citations, or references, to published academic legal writings in superior court decisions is a complex and challenging task. It requires in-depth knowledge of legal terminology, language, and the ability to comprehend complex legal texts. The process involves identifying references to previous court decisions, statutes, regulations, and other legal materials to help understand and interpret the current case.

In Portuguese legal decisions, it is common to cite, or even quote, specialized works that provide relevant context towards the decision being issued. However, the lack of a consistent format on how citations or quotations are written in official documents makes automatic detection a difficult problem.

1.1 Context and Motivation

Detecting references to legal literature within superior court decisions and crediting them to the authors being cited is important for legal professionals, scholars, researchers, authors pursuing an academic career [1], and to reviewers who, for example, want to find all court rulings that have cited a particular work, or determine in which contexts a particular work has been cited.

The identification of references plays a critical role in the development of legal technology and the improvement of legal research tools. With the increasing volume of legal texts available online, it is essential to have accurate and efficient reference identification tools to support legal professionals in their work. In the era of big data, the sheer volume of legal documents can be overwhelming without the proper tools to help in parsing and analysing them.

A reference identification tool can have a good impact on the justice system as a whole. By facilitating the identification of relevant legal literature, it assists judges in locating the sources that have been consulted to support a specific solution. Thoroughly considering prior decisions and the accompanying citations contributes to the unification of rulings and promotes increased certainty and security in the application of the law. Additionally, it may contribute to the

development of legal arguments, the settlement of legal disputes, and progress in the rule of law.

This work aims at developing innovative text mining-based solutions to automatically identify citations in the official court ruling documents and credit them to the authors being cited, while also storing them and indexing the cited author so that the data can later be visualized by the users.

The work presented is developed using real legal documents gathered from DGSI's website [2], from the *Ministério da Justiça* (Portuguese Ministry of Justice).

1.2 Main goals

In order to tackle the automatic identification of citations and to allow a user to visually observe court rulings citing their work, the following goals guide the work of this dissertation.

- **Language-focus:** The focus of this work is on parsing Portuguese superior court decisions, written in the Portuguese language, to determine citations to legal literature.
- **Data availability:** In order to develop and train an identification algorithm, allow for the available documents to be stored in the same computation environment, together with the link to the original document.
- **Identification algorithm.** Develop an algorithm that uses real superior court ruling documents to learn, with high accuracy, how to identify citations within those documents.
- **Citation extraction:** Citations identified from sentences need to be parsed in order to produce the corresponding publication metadata, which is normally characterized by elements such as authors, publication title, pages and year.
- **Indexing citations:** In order, for authors to receive proper credit, it is essential to appropriately link a citation to both the document in which it appears and the corresponding author.
- **Authors' profile:** Authors must be credited with the works they have authored and citations their works are receiving in court decisions. This enables the ability to a given author to fetch citation to their own documents.

In order to accomplish these goals, the work must include the following necessary steps:

- **Data preprocessing of superior court decision documents.** In order for these documents to be used and their information to be easily accessed by an algorithm, one needs to pull them from a certified and official website and perform some cleanup and syntactic analysis before storing them in a database.

- Citations need to be carefully looked at as they are not always referenced the same way and documents may use more than one type of referencing.
- The algorithm not only needs to accurately determine the parts of the document that correspond to a citation, but also to interpret it and discover its structure made of authors, title, where-published, and year, in order to store it properly in the database. The set of citations, or references, identified in a document constitutes its references list.
- Plan the storage model to be concise yet future-proof. Its information should be accessed and used in multiple interesting ways beyond this dissertation.
- Create a mockup project and implement a prototype that illustrates the algorithm in practice, namely the citations it discovers in a document, and the citations an author has received in each of his works.

Based on the formulated goals, the following research questions guide the work pursued in this dissertation:

- **Q1:** How to create an algorithm that selects citations, or quotes, in a Portuguese text through collected superior court decision documents with a diversity of citation syntax?
- **Q2:** Can this identification process benefit from Machine Learning ([ML](#)) capabilities?
- **Q3:** How to credit researchers and law practitioners with citations to their work found in superior court decision documents?
- **Q4:** How to identify the metadata elements characterizing a citation in order to build a list with all references associated with a specific work?

Figure [2.1](#) illustrates the different stages of the project addressing the main goals, forming a pipeline.

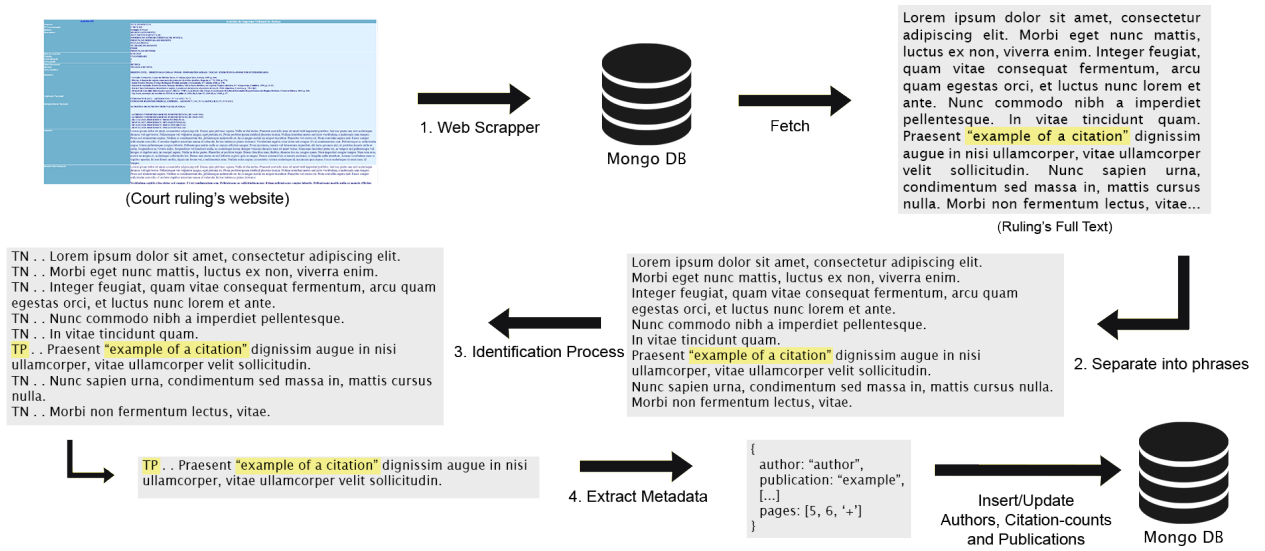


Figure 1.1: Project's Pipeline

1.3 Contributions

This work resulted in contributions and tools to the scientific community in the following way:

- Our thesis introduces an algorithm that utilizes state-of-the-art Natural Language Processing (NLP) technology to accurately identify citations within legal portuguese literature. This innovative solution significantly reduces manual effort by automating the reference identification process with exceptional precision. Moreover, the algorithm organizes the extracted metadata into a structured database, providing a valuable resource for future research and presentations. By establishing a solid foundation, our work opens avenues for advancements in the field of reference identification from legal literature.
- In addition, we have developed a fine-tuning to BERT's transformer architecture, with legal field content. Its potential lies in enabling more accurate and context-aware citation recognition, potentially revolutionizing the way legal references are identified and processed. By leveraging advanced NLP and ML techniques, our work contributes to the efficiency and accuracy of reference identification in the legal domain.

1.4 Dissertation structure

The remainder of the dissertation is organised as follows:

- In Chapter 2 is presented essential background information for the broader topics of the dissertation and some techniques or technologies state-of-the-art related to them;

- Chapter 3 presents the steps of data preparation, explaining the dataset used and its transformations for later use;
- The initial project development starts at Chapter 4 where we explore how the reference identification is made;
- Chapter 5 focuses on metadata extraction from the references identified for later use;
- Chapter 6 uses information gathered throughout the work to elaborate demonstrations on different topics;
- The dissertation concludes on Chapter 7 and there is presented an idea of future work.

The dissertation's pipeline, depicted in Figure 1.2, will be morphed when we reach each one of those stages throughout the dissertation.

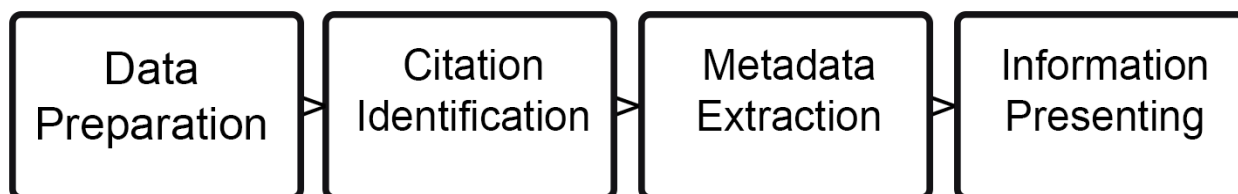


Figure 1.2: Dissertation's Pipeline

Chapter 2

Background and Related Work

This chapter focuses on introducing relevant concepts related to this dissertation, namely to the areas of Data Mining (DM), Text Mining (TM) and Natural Language Processing (NLP) and presenting some techniques and data tools that are related to our work, as well as describing related research and tools addressing similar problems. It also explores the state of the art on NLP technologies, followed by a more minute enlightenment on transformer architecture, as this is, by our understanding, the most relevant form of data exploration techniques in nowadays machine learning technology. After traversing through NLP, the chapter works around Citeseer and its legacy in the text identification field. This chapter ends by stating Non-relational databases and their different uses.

2.1 Data and Text Mining

Data mining is a complex process that involves discovering patterns and insights from extensive datasets using statistical, machine learning, and data visualization techniques. Its primary objective is to extract valuable information to support decision-making and predictive modelling. The IEEE defines data mining as "the extraction of knowledge from data, through the use of statistical, computational, and visualization techniques." [3]. This versatile approach finds applications in various fields, including marketing, healthcare, finance, and cybersecurity.

The data mining process comprises several essential steps. Initially, data is collected from diverse sources such as databases, social media, and websites. Subsequently, the collected data undergoes preprocessing operations, including cleaning, filtering, and transformation, to ensure its suitability for modelling. The modelling phase involves the application of statistical and machine learning algorithms to uncover hidden patterns and relationships within the data. Common techniques employed in data mining encompass clustering, classification, regression, and association rule mining. Rigorous evaluation and validation of the modelling results are conducted to verify the accuracy and usefulness of the derived models.

While data mining offers numerous benefits, such as identifying new business opportunities

and enhancing decision-making processes, it is not without risks. Privacy violations can arise as a result of accessing and analyzing large datasets. Additionally, the potential for biases and discrimination in decision-making processes must be carefully considered. Despite these challenges, data mining continues to play a crucial role in extracting valuable insights from data, empowering organizations in various industries to make informed decisions and drive innovation.

Text mining is a valuable process that extracts useful information and insights from unstructured text data using Natural Language Processing (NLP) techniques. Its primary aim is to uncover patterns and relationships within text data, which can be utilized for decision-making or predictive modelling. The IEEE defines text mining as "the application of natural language processing, information retrieval, and machine learning techniques to extract information and insights from unstructured text data." [3]. It finds applications in various domains, including sentiment analysis, topic modelling, and document classification.

Text mining involves several crucial steps. Initially, data acquisition takes place, wherein text data is gathered from diverse sources such as websites, social media, and documents. Subsequently, the acquired data undergoes preprocessing, which includes tasks like tokenization, stop-word removal, and stemming. These preprocessing steps are crucial in transforming unstructured text data into a format suitable for modelling and analysis.

The modelling phase in text mining entails the application of NLP and machine learning algorithms to preprocessed text data. These algorithms enable the discovery of patterns and relationships within the text. Common text mining techniques include sentiment analysis, topic modelling, and named entity recognition. The results of the modelling process are evaluated and validated to ensure accuracy and usefulness. It is also essential to be aware of potential risks associated with text mining, including privacy violations and the potential for bias and discrimination in decision-making processes.

To this work in particular text mining is beneficial as it can help extract reference information to create the document bibliography and to identify patterns in the Portuguese language with the help of NLP techniques.

2.2 Data Mining Pipeline

In the modern world, the application of DM is frequently planned using a pipeline of operations. The sequence of these operations makes it possible to convert information input into knowledge output.

As stated by Juodyte M. in "Overview: Data mining pipeline." [5], the CRISP-DM model grew out of an earlier model named Knowledge Discovery in Databases (KDD). It is also important to note that this pipeline is iterative, allowing us to go back and repeat previous steps in order to strengthen the data output produced at the end. The CRISP-DM model, which can be seen in figure 2.1, is currently the procedure that is most widely used in the community and it involves

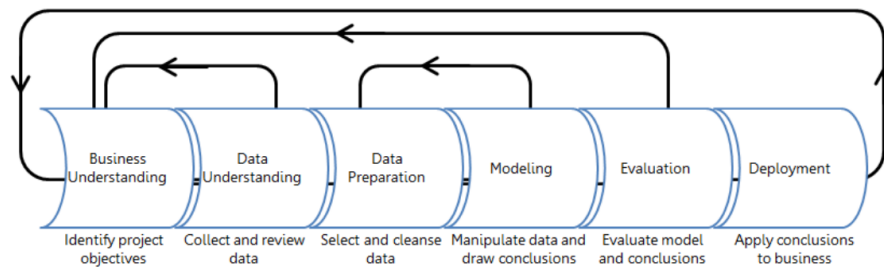


Figure 2.1: Cross Industry Standard Process for Data Mining (CRISP-DM)[4][5].

the following phases:

- Business Understanding.** This is a phase that entails comprehending the project's boundaries and goals from a comprehensive and distributional point of view. As a result, before planning how the outlined objectives will be attained, it should be first defined a [DM](#) problem. It is up to the team working with the data to define this progress in coordination with the subject, never forgetting the resources, tools, techniques, risks, and costs. However, even in situations where the business is differentiated, as is the case in law, there will always be a problem, a plan to follow, and conclusions to be drawn from the work.
- Data Understanding.** It implicates the entire environment of data collection-related activities, as well as those that help data scientists become familiar with the information's quirks, data quality issues, and initial approaches and hypotheses about what they have. When used in the following stage, data that is incorrect and poorly organized is fairly typical.
- Data Preparation.** This involves all the steps necessary to obtain a final dataset that is ready to handle, including the application of other tools for data inclusion and exclusion known as data selection as well as the selection of tables, records, and attributes. In a data cleaning phase, the dataset is carefully inspected for incorrect values and inconsistencies in data types and formats. This phase's goal is to make it possible to statistically analyze the data using tools that need the data to be properly formatted, normalized, and regulated.
- Modelling.** During this phase the parameters are changed to produce the best results while the techniques and procedures to be used on the data are chosen. Different modelling techniques can be used, including anomaly detection to identify outliers and abnormal values, association rule learning to simulate dependent relationships between variables, clustering to identify groups and subgroups of similar values, and classification to apply previously learned knowledge to new data and categorize it. As opposed to summarization, which aims to create a concise visual representation of the dataset, regression algorithms typically infer the relationship between a dependent variable and independent variables.
- Evaluation.** In the Evaluation phase, a quality model must be tested to see if it meets the requirements imposed in the previous phases. Although deployment is the final step, the

pipeline is constantly active because most hypotheses can always be improved and made applicable to modern standards by new information.

2.3 Data Preprocessing and Preparation

According to Jiawei Han et al. [6], data preprocessing aims to assess and arrange factors that endanger data quality, such as accuracy, completeness, consistency, timeliness, believability, and interpretability. Incorrect, inconsistent and incomplete data are integral properties of currently collected data that are part of real data databases and data warehouses with exceptionally strong ties to their genesis's everyday living. Additionally, raw logs, documents, and semi-structured data are commonly in a form that is not required for processing.

It is worthwhile introducing the concept of concealed missing data, even though the existence of incorrect data can be simply explained by human or computational error. A user may occasionally decide to enter a default value or a wrong value in one of the information that concerns him for a variety of reasons, according to Pearson [7]. This also applies to the fields of law or legal data, as academics or law practices may voluntarily enter information that is not accurate (such as a person's name or particular information that may expose an individual). Although such information is fundamentally incorrect, an automated system accepts it as long as it is entered in accordance with its formatting guidelines. This kind of data distorts results and is inherently challenging to comprehend, identify, and remove.

Observed evidence suggests that incomplete data is often entered due to accessibility issues or entry errors. The quality of data is also influenced by how consistently it is collected. It is crucial to evaluate the data's reliability on references and its interpretability, which measures how the information is perceived. If a phrase presents a reference that might not be clear or with the objective of being tacit, it can create increased complexity in trying to figure out if something is being referenced.

Data cleaning, data integration, data reduction, and data transformation are some of the key tasks that must be considered during data preprocessing.

The goal of the data cleaning phase is to address issues such as missing values and data noise. This phase also addresses the presence and identification of outliers, which may or may not be the goal of the work, as well as dealing with data inconsistencies. We should keep in mind that these procedures must be reliable to prevent over-fitting the data and making it challenging to obtain accurate results.

Examples of some of the issues and mistakes that can occur during data collection include:

- Technical problems with sensors and the hardware restrictions they have on data collection and transmission
- Scanner errors brought on by imperfect recognition methods (e.g. speech-to-data).

- The refusal or preference to conceal personal information due to user privacy concerns.
- Manual errors and manually created data.
- The responsible entity might not be able to collect certain types of data, due to their implications or cost.

During this phase, encountering missing values poses a significant challenge, necessitating careful consideration of appropriate solutions. The entire record with missing values could be ignored, or they could be manually filled in or replaced with constants (e.g. "Unknown." or "NaN"). Utilizing attribute trend calculations, such as the mean or median, from the entire dataset or from particular classes or subsets within it is another choice. Researchers can also choose to use the most likely value to complete the missing data. Despite their widespread use, it's crucial to remember that these latter techniques have the potential to introduce biased and inaccurate values into the dataset. The method of estimating missing values is referred to as "imputation". As a result, by providing approximations for the missing values, imputation plays a crucial role in reducing the impact of missing data and enabling meaningful analysis.

Regarding the outliers found in the data, these may have been found through clustering, which groups values into clusters based on shared traits. Figure 2.2 shows an example of an outlier, a record that deviates from the norm within these characteristics, and the use of metadata, knowledge about the data itself, a priori, can be helpful in confirming the existence of outliers.

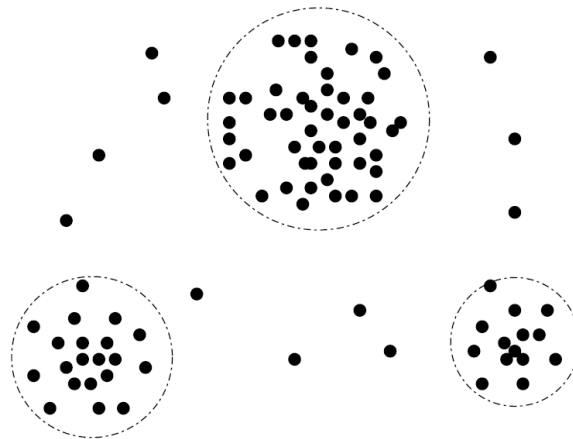


Figure 2.2: A 2-D customer data plot with respect to customer locations in a city, showing three data clusters. Outliers may be detected as values that fall outside of the cluster sets.[8].

Cross-referencing data from various sources is a concept known as data integration, which is used to gather data from various databases, cubes, or files and represent a concept more broadly. Additionally, by avoiding redundant data and inconsistent results in the final dataset, this process speeds up the [DM](#) process that follows.

2.4 Non-relational Databases - NoSQL

Not only SQL (**NoSQL**) refers to a class of non-tabular databases designed to manage large amounts of data that are not properly suited to the conventional relational database model. They instead make use of a range of data models created to support specific use cases. MongoDB, Cassandra, and Redis are a few well-known examples of NoSQL databases.

2.4.1 Types of Non-relational Databases

The different types of NoSQL databases include:

- Document databases: Data is stored in the form of documents, which are comparable to JSON objects. Due to their ability to store complex data structures and support flexible schemas, document databases are frequently applied in modern online applications.
- Key-value databases: These types of databases store data as a collection of key-value pairs, with the key being used to look up the corresponding value. Although they are quick and effective, sophisticated searches cannot be run on key-value databases.
- Wide-column stores: Designed to read and write enormous amounts of data quickly and efficiently as they store data in columns rather than rows. Wide-column stores are often used for big data applications.
- Graph databases: Used to represent complex data-to-data relationships since they store data in the form of nodes and edges. Graph databases are often used for recommendation engines and social networking applications.

NoSQL databases are frequently chosen over typical relational databases when the data model is not well understood beforehand, when the data is big and/or unstructured, or when scalability is the main concern. For applications that demand complex operations or strict reliability guarantees, they might not be the optimal choice.

2.4.2 Different cases of NoSQL databases

The following items are three of the most popular database programs [9], each one with its unique capabilities:

- MongoDB is a document-oriented database that stores data in JSON-like documents with optional schemas. It is frequently utilized in modern web applications because of its scalability and flexibility. It provides flexible schemas, making it simple to add and modify data over time, it's horizontally scalable, meaning that it can handle high volumes of data and traffic by simply increasing the number of servers in the cluster but it doesn't support complex transactions.

- Cassandra is a wide-column storing database that is designed to handle large amounts of information across many commodity servers, providing high availability with no single point of failure. It can replicate data across servers to improve availability and fault tolerance. Cassandra also allows users to choose the level of consistency that is appropriate for the application in the case. It may require upfront planning and design, and a more complex to perform some queries compared to other NoSQL databases, doesn't support complex transactions. It's normally used in real-time analytics and online advertising.
- Redis is an in-memory data structure store that can be used as a database, a message broker, and a cache. Although it is not a traditional NoSQL database, its way of reading and writing massive amounts of that allows it to be used as such. It supports a wide range of data structures, going from lists to sorted sets and has a publish-subscribe messaging, which allows clients to subscribe to channels and receive messages in real time. Redis' high-performance flexibility makes it more commonly used for message brokering, real-time analytics and caching.

2.5 Natural Language Processing and Large Language Models

NLP's origins can be traced back to the 1950s [?] when researchers started creating techniques to make it easier for machines to understand human language. Early methods made use of syntactic and grammatical patterns and rule-based systems. But these methods frequently fell short of capturing the subtleties and complexity of natural language, necessitating a move towards more data-driven strategies.

The introduction of large language models (LLMs) signalled a significant turning point in NLP research [10]. These models, distinguished by their enormous size and impressive computational power, are able to process and produce text at a scale that has never been seen before. They make use of highly developed deep learning architectures like Transformers, which are excellent at capturing long-range dependencies and contextual relationships within language.

Models like GPT-3.5 show remarkable performance in generating pertinent and cogent responses when it comes to information retrieval and question answering [11]. Clinical decision-making, automated diagnosis, and analysis of electronic health records are all made possible in the healthcare industry by NLP. Large language models have also been useful in text summarization, sentiment analysis, machine translation, and even creative writing.

Despite their enormous potential, large language models pose ethical challenges. Concerns about privacy and data protection are brought up by the sheer volume of training data. Additionally, these models have been shown to amplify biases found in the training data, increasing the likelihood that social, racial, and gender disparities will persist. To ensure the ethical and impartial use of NLP models, these ethical conundrums must be addressed.

The development and deployment of large language models present a number of technical

challenges. Such models require a lot of computational power to train, which makes them inaccessible to smaller research teams [10]. Additionally, ongoing research directions include reducing biases, enhancing interpretability, and boosting model generalization. Additionally, finding a balance between model capability and control over generated text is still an area of research.

Large language models have enabled Natural Language Processing to advance in a way that has never been possible before. These models have revolutionized many applications across domains by changing how machines understand and produce human language. However, ethical issues and technical difficulties continue, calling for ongoing research and responsible application. NLP is the current best technique to use machines to develop and enhance human language in many known and unknown fields of computer science that involve natural language.

2.5.1 NLP Techniques

A wide range of applications, including document classification, sentiment analysis, spam detection, and more, are now possible thanks to recent significant advancements in named-entity recognition techniques and technologies. To produce cutting-edge results, these methods make use of the strengths of machine learning, NLP, and deep learning. We will talk about some of the popular methods and tools for named-entity recognition here.

- **Tokenization:** This is a fundamental preprocessing step that involves breaking down a text into smaller units, called tokens. These tokens can be individual words or subwords, depending on the chosen tokenization strategy. Tokenization forms the foundation for various downstream tasks, such as text classification, named entity recognition, and machine translation. By dividing the text into tokens, NLP models can effectively process and understand the language, as each token represents a discrete piece of information. Additionally, tokenization also helps in handling challenges like handling punctuation, special characters, and formatting inconsistencies in the text. As a cornerstone of NLP, tokenization enables language models to analyze, interpret, and generate human language, playing a vital role in the success of many NLP applications and advancements in the field.
- **Word Embeddings:** Word embeddings are detailed vector representations that reveal the connections between words' semantic meanings. Text identification tasks have been performed better thanks to methods like Word2Vec[12], GloVe[13], and FastText[14]. These embeddings aid in better text generalization and comprehension by assisting models in recognizing contextual and semantic similarities between words.
- **PoS Tagging:** Part-of-Speech (PoS) Tagging is a crucial linguistic analysis technique that involves assigning grammatical labels or tags to each word in a given text, based on its role and function in the sentence. The PoS tags represent parts of speech, such as nouns, verbs, adjectives, adverbs, and more. This process is essential for understanding the syntactic structure of sentences and extracting valuable information about how words

relate to each other. PoS tagging is a challenging task due to the inherent ambiguity of natural language, where a single word can have multiple meanings and functions depending on the context. However, accurate PoS tagging is fundamental for many NLP applications, such as text parsing, machine translation, and sentiment analysis. It enables language models to disambiguate words and gain a deeper understanding of the underlying grammar, leading to more sophisticated and context-aware language processing capabilities. As a result, PoS tagging significantly enhances the accuracy and performance of various NLP systems, empowering them to tackle complex language-related challenges effectively.

- **Chunking:** is a crucial technique in syntactic analysis that involves grouping individual words in a sentence into meaningful and well-defined units, called "chunks." These chunks often consist of phrases or segments, such as noun phrases, verb phrases, and prepositional phrases. This technique is a vital step in understanding the grammatical structure of a sentence and plays a significant role in various NLP applications, such as information extraction, text summarization, and sentiment analysis. By breaking down sentences into chunks, NLP systems can extract essential information and relationships between words, leading to a more accurate and comprehensive understanding of the text. Through this process, NLP chunking empowers language models to handle more complex tasks and aids in the development of sophisticated natural language understanding systems.
- **Pretrained Language Models:** Due to their capacity to extract representations from sizable text corpora, pre-trained language models have become extremely popular. Massive amounts of data have been used to train models like BERT, GPT-3, and their derivatives, which can be tailored for particular text identification tasks. To achieve superior performance in tasks like text classification, sentiment analysis, and named entity recognition, these models capture nuanced language aspects and context.

2.5.2 NLP Technologies

In the area of text identification, there are a number of well-known technologies and libraries, particularly in the Python ecosystem, that offer strong tools and frameworks for creating text recognition solutions. Here are a few noteworthy ones:

- **Natural Language Toolkit (NLTK):** NLTK is a well-known Python library for NLP tasks. Tokenization, stemming, tagging, parsing, and classification are just a few of the tasks it can perform, and it offers a complete set of tools and resources for them. NLTK is a useful tool for text identification research and experimentation because it also has a large number of corpora and lexical resources.
- **TensorFlow and Keras:** TensorFlow is a robust deep-learning framework that is frequently used to create neural networks, including those for text recognition. Building and training deep learning models are made easier by Keras, a high-level API built on top of TensorFlow. For tasks like text classification, named entity recognition, and text

generation, these libraries provide tools for building Recurrent Neural Network ([RNN](#)), Convolutional Neural Networks ([CNN](#)), and transformer-based models.

- **PyTorch:** PyTorch is a further well-liked deep learning framework that is renowned for its adaptability and dynamic computation graph. It offers a robust ecosystem for creating neural network models for tasks involving text identification. To experiment with the most recent developments in NLP and text recognition, researchers and developers can use PyTorch's modules and utilities for building [RNN](#), [CNN](#), and transformer-based models.
- **spaCy:** spaCy is a well-liked Python library for effective and powerful NLP. It offers named entity recognition, part-of-speech tagging, dependency parsing, and tokenization that is both quick and accurate. Additionally, spaCy provides word vector embeddings and supports customized model training for jobs like entity and text classification.

Another important library is Hugging Face Transformers, which provides pre-trained models for Transformer models. Hugging Face Transformers provides cutting-edge models for tasks like text classification, question answering, entity recognition, and text generation.

It is significant to note the following tools/libraries, while being relevant and modern, weren't utilized because they did not support Portuguese sentences:

- Gensim [\[15\]](#), that should be replaced with PyTorch and BERT when used in Portuguese, for better results
- Flair [\[16\]](#), that is still in development at the time of writing, but is very promising
- GROBID [\[17\]](#), that only can be used in PDF files, as it works with layout tokens.

2.6 Transformer architecture

[NLP](#) has been an active area of research for decades, and the recent explosion of data and computing power has accelerated progress in the field. The Transformer architecture, introduced by Vaswani et al. in 2017 [\[18\]](#), has been one of the most important developments in NLP in recent years. The Transformer architecture has quickly established itself as a pillar of contemporary NLP, powering cutting-edge models for activities like language modelling, text classification, and translation. [\[19\]](#)

This architecture is based on the self-attention mechanism, which enables it to capture long-range dependencies in input sequences. This approach has shown to be very successful for modelling languages because it enables the Transformer to pick out the most important information from the input and use it to generate predictions.

The Transformer architecture has transformed NLP and created new opportunities for language creation and comprehension. To understand this architecture's advantages and disadvantages, it

is possible by examining its internal workings, and by working to create NLP models that are even more potent as a result.

2.6.1 BERT - Bidirectional Encoder Representations from Transformers

Bidirectional Encoder Representations from Transformers ([BERT](#)) was developed by Google in 2018, and is based on the Transformer architecture in the paper "Attention is All you need" by A. Vaswani et al. in 2017 [18]. It is a pre-trained language model that has been trained on a massive amount of text data, including Wikipedia articles and the BookCorpus dataset [20].

The pre-training process consists of two steps:

1. the model is trained on a masked language modelling task where it predicts the missing word in a sentence given the context
2. the model is trained on a next sentence prediction task where it predicts whether two sentences are consecutive in a text or not.

2.6.1.1 Architecture

[BERT](#) consists of a stack of Transformer encoder layers, each of which has two sublayers: a position-wise feedforward network and a multi-head self-attention mechanism. The feedforward network applies a non-linear transformation to the output of the self-attention mechanism, allowing the model to attend to different elements of the input sequence and capture the relationships between them.

One of [BERT](#)'s key characteristics is its bi-directionality, which enables the model to consider both the left and right contexts of a given word in a sentence. In contrast, unidirectional training was used to develop earlier language models like GPT or semi-bidirectional models like ELMo. [21] (figure 2.3)

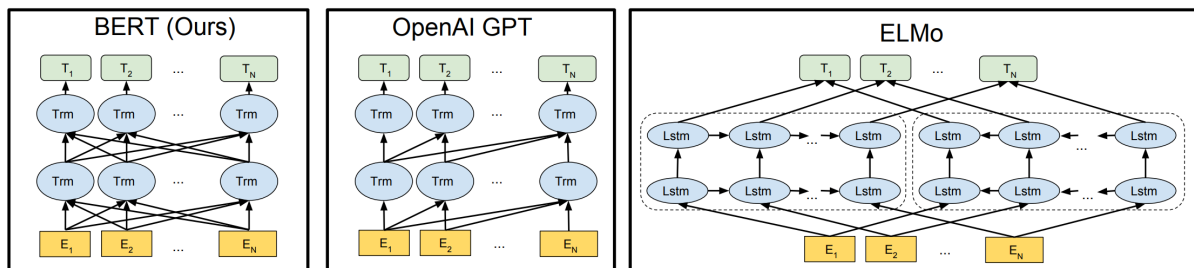


Figure 2.3: Comparison of BERT, OpenAI GPT and ELMo[20].

2.6.1.2 Applications

On a range of NLP tasks, such as question answering, sentiment analysis, and natural language inference, BERT has achieved cutting-edge performance. The General Language Understanding Evaluation (GLUE) benchmark, which consists of nine various tasks that test the model's capacity to comprehend and produce natural language, is one of the most well-known benchmarks for assessing NLP models.

BERT has achieved state-of-the-art performance on other benchmarks, such as the Stanford Question Answering Dataset (SQuAD) and the Conference on Computational Natural Language Learning (CoNLL) datasets, and has outperformed humans on some of the GLUE benchmark's tasks [20]. Figure 2.4 presents an idea of how the input is treated by BERT in masked language model.

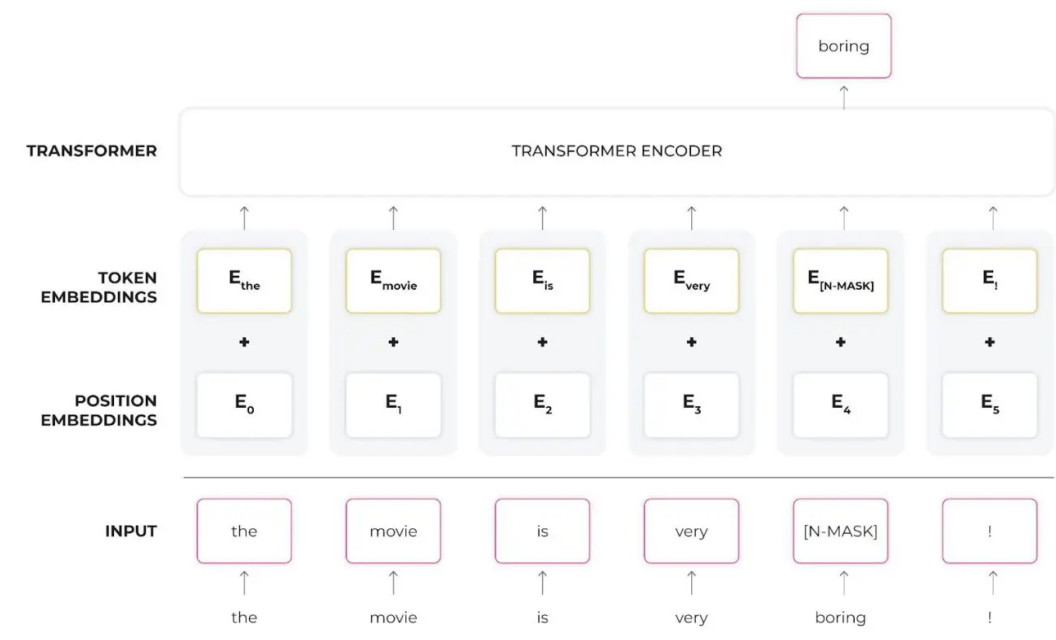


Figure 2.4: Model structure of the label-masked language model. "N-MASK" is a mask token containing negative label information.[22].

2.6.1.3 New Products and Conclusion

BERT has transformed the field of NLP by demonstrating the power of pre-trained language models that can learn from massive amounts of data. The model has been able to capture the relationships between various sentence components and achieve cutting-edge performance on a variety of tasks thanks to its bidirectional nature and attention mechanism.

BERT's innovation in 2018 made Google launch a new product called Language Model for Dialogue Applications (LaMDA) in 2021 and, as the name stands for, the application was made as an open-ended conversational AI application. This type of Transformer, to be able to do

its objective, was fine-tuned with conversation datasets. Later, in 2023, Google integrated this application into a new project, called BARD, a competitor of ChatGPT-4 and a conversational, AI chat service that uses Google's search engine capabilities to gather its information and deliver [23].

2.6.2 GPT - A State-of-the-Art Language Model

The paper "Improving Language Understanding by Generative Pre-Training" by Radford et al. from OpenAI,[11] introduced the Generative Pretrained Transformer (GPT) architecture. GPT is a type of neural network that produces natural language text using a transformer-based architecture. To achieve cutting-edge performance, BERT is first pre-trained on a large corpus of text data using a language modeling objective. After pre-training, it is fine-tuned on downstream NLP tasks to adapt it for specific applications and achieve high performance.

2.6.2.1 Architecture

The architecture of GPT consists of a stack of transformer decoder layers, where each layer transforms the input into a series of hidden states using multi-head attention and feed-forward networks. The output sequence of tokens is produced by the final layer by sampling from a probability distribution over the vocabulary. [24]

2.6.2.2 Applications

GPT has been applied to a variety of NLP tasks, such as question-answering, text summarization, and language translation. The model has produced high-quality text with state-of-the-art performance in several benchmark datasets, proving its efficacy.

Additionally, GPT has been applied to the creation of chatbots and virtual assistants, where it has demonstrated promise in producing contextually sensible answers to user inquiries.

It received its popularity not only from being state-of-the-art but from the free-to-use chatbot ChatGPT [25] and from the image generator from input, DALL-E [26], both created by the same company as GPT. ChatGPT uses GPT-3.5 technology and its viral sensation set the record for the fastest-growing user base [27].

2.6.2.3 GPT-4 and Conclusion

GPT has become a significant step forward for NLP and has shown how effective big language models are at producing high-quality text.

GPT-4 is, at the time of this writing, the next iteration of GPT. It surpasses GPT-3.5 with flying colours. It is 40% more likely to produce factual responses than its predecessor [28]. It

ends up being more coherent in its responses, has a better understanding of complex problems, reads and writes programming with less error margin, and, as the major novelty by merging DALL-E into the chatbot, interprets given images and graphics. [29] GPT-4 and its API it still, at the time of this writing, in beta testing and can only be used by a subscription fee to OpenAI's service.

In a variety of NLP applications, its capacity to produce coherent and contextually appropriate text has demonstrated promise, and its influence on the field is likely to continue in the years to come.

2.6.3 Other Models and Applications

Other interesting models that large-scale corporations are working on:

- PaLM by Google: a model introduced to do more than being a chatting and searching tool. This model aimed to be a collaborative project between language models and a helper robot that would do tasks such as solve complex math word problems, answer questions in new languages and explain jokes. [30] [31]
- LLaMA by Meta AI: A Large Language Model (LLM) made by Meta's AI team to better serve its chatbots and virtual assistants to content creation and data analysis

All of these models are being each day more used by huge applications like Microsoft in its Bing AI [32], Duolingo [33] and Salesforce [34].

2.7 Related Work

To fully grasp the available work done previously in context of this dissertation, multiple sources were verified. We look into work related the various steps of our pipeline, such as citation identification [35] [36], text summarization in legal literature [37], works on the same effort but for portuguese solutions using BERT [38], network architecture for citation in documents [39], and many more in the gray area between those fields. The most important piece of investigation that seems to be correlated with all of the researched documents is Citeseer.

2.7.1 Citeseer - An Automatic Citation Indexing System

Citeseer [36], is an influential digital library launched in 1997 that has transformed the way academic papers are accessed and explored by researchers by establishing itself as a crucial resource for academics all over the world thanks to its extensive collection of scientific articles. Citeseer came into existence at a time when the internet was becoming more and more popular as a platform for sharing research. For researchers, the rapid expansion of digital repositories

brought both opportunities and difficulties. Libraries and physical journals, which were once sufficient sources of scientific literature, were becoming less so. CiteSeer was developed in response to the demand for a complete, effective, and easily accessible digital library.

With the use of cutting-edge technologies, CiteSeer was created as an autonomous digital library that automates the gathering, indexing, and organizing of scholarly publications. It navigates the internet using web crawling techniques to find academic papers and extract pertinent metadata. The distributed system and multiple servers that makeup CiteSeer's architecture allow for effective paper indexing and retrieval. Its search engine uses a sophisticated ranking algorithm to make sure that pertinent and excellent articles are given priority.

A wide range of features and functionalities are available in CiteSeer to improve the research experience for scholars. First off, its extensive collection covers a variety of academic fields, such as computer science, physics, and mathematics, among others. From a single platform, researchers can access a wealth of data, saving a significant amount of time and effort. Additionally, CiteSeer offers citation analysis, which enables users to follow the influence and impact of particular articles within the academic community. Additionally, it provides user-specific recommendation systems that make recommendations for related papers based on search preferences and history.

Despite its numerous benefits, CiteSeer does have some limitations and constraints. The incomplete coverage of academic papers is one apparent drawback. Despite CiteSeer's best efforts, it is not possible to index all articles due to the enormous amount of research that is conducted on a global scale. Furthermore, because CiteSeer relies on automated indexing algorithms, metadata extraction and search results may occasionally be inaccurate. The depth of analysis that can be carried out within the platform is also constrained by the absence of sophisticated text mining and natural language processing techniques.

2.7.2 CiteSeerX

CiteSeerX [40] was introduced to enhance the research experience made in CiteSeer and brought significant advancements and expanded its scope beyond computer science. One of the notable advancements was the addition of new disciplines, which covered a broad spectrum of scientific disciplines like physics, biology, and social sciences. The platform's utility was increased by this expansion, which gave researchers access to a wide range of scholarly literature from various fields. This included the law field.

In order to enhance search functionality and user experience, CiteSeerX introduced a number of changes. More precise and pertinent search results were produced as a result of improved search engine algorithms. High-quality papers were ensured to be prominently displayed in the search results by means of sophisticated ranking algorithms. The updated user interface provided by CiteSeerX also made it easier to use and more logical. The enhanced interface simplified accessing and exploring scholarly publications and allowed for easier navigation.

CiteSeerX's incorporation of improved metadata extraction methods led to more precise and

thorough metadata for indexed papers. The extraction procedure was improved to include crucial information like author affiliations, abstracts, and citation data. The platform's usability and comprehensiveness were greatly enhanced by this change. Additionally, CiteSeerX advanced citation analysis further by enabling researchers to investigate citation networks, monitor citation trends and assess the significance of research papers across a range of disciplines.

As digital libraries increased in size and complexity, scalability and performance issues surfaced, CiteSeerX addressed these issues. The distributed architecture it introduced made it possible to efficiently index and retrieve papers from various servers. This architecture made it easier for the system to manage large amounts of data, ensuring that users encountered few delays when accessing search results. CiteSeerX's scalability and performance enhancements were essential in meeting the demands of an expanding user base as well as the growing volume of research literature.

In conclusion, by following CiteSeerX's crucial evolution from CiteSeer, it could be possible to understand how to improve the code to a more scalable and reliable version. As detailed in the next chapters, CiteSeer was a great base to follow as a proof of concept and CiteSeer was a great addition to the knowledge to have some guidelines on where to look to improve the work throughout its development.

2.8 Summary

This chapter summarizes fundamental concepts related to the work done and that is going to be detailed in the following chapters. It begins by explaining briefly some [DM](#) concepts regarding text mining, its pipeline, and with an extra focus on the data preprocessing phase that uses tools to make data cleaning, data integration, data reduction and data transformation. The chapter concludes on developing around [NLP](#) and [LLM](#), provided an introduction to concepts related to them, and explains, in a brief manner, the transformer architecture, as it is a crucial part of the [NLP](#) state-of-the-art and it is used in the project. Here was also possible to understand the current state of NLP technologies and techniques, Transformer architecture as a concept and current applications of it, and explore non-relational databases that were available to work on the project.

Chapter 3

Data Preparation and Preprocessing

As described in the previous chapter, namely for CiteSeer’s and other similar projects, any approach to identify citations in a document needs to start by collecting and preparing a dataset to be properly curated, used and tested accordingly [36] [40]. This chapter describes the initial steps taken in our project to create a collection of superior court decision documents, their preprocessing and database storing aiming to simplify the learning task for the identification algorithm being developed.

As we can see in Figure 3.1, this chapter starts by describing a tool created to extract court rulings documents from a public Web repository, parsing and special characters cleaning, and organizing them into structured files which are store in a local database for future use.

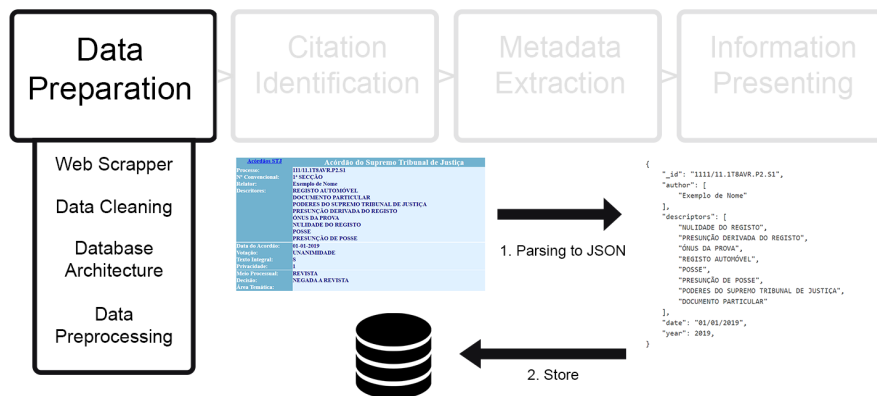


Figure 3.1: Project’s Pipeline: Data Preparation Stage

3.1 Data Gathering - Web Scrapper

Considering that the DGSI’s website does not support an API to access the published law court documents, we developed a web-scraper tool to extract them. Given that these documents are freely available on the website without apparent rate limits, we were compelled to utilize

this approach due to our specific research requirements. The tool was created to be effective, scalable, and able to handle a significant amount of data. It carefully searches and extracts from the website pertinent superior court decision documents. The scraper includes error-handling features to deal with unforeseen circumstances and it is optimized to use parallel methods or asynchronous requests for quick data retrieval. This tool can be easily modified to allow future researchers or developers to adapt it to their needs.

An initial selection of twenty links to documents known to include citations to a set of known authors was gathered. Furthermore, they were deliberately chosen because they make a good sample of the diversity of layouts, writing procedures, and reference styling. Each link document presented unique characteristics that differentiated it from the others in terms of structure, content presentation, and citation format. By incorporating this diversity, we aimed to capture a comprehensive and representative sample, enabling a thorough analysis and evaluation of the varying styles and approaches employed in the respective sources.

3.1.1 Technologies Background

We chose Python as the scripting language for developing the web scrapper due to its simplicity and extensive library support, which made it well-suited for the task. The script utilized the "request-html" library, specifically the "HTMLSession" method, to send an HTTP request to the DGSJ's URL. To ensure the request was handled smoothly and mimicked a browser, a specific header was included. The library "re" was also employed, particularly during this stage, to identify patterns in the data. Additional details can be found in Table A.1.

Web scrapping normally is used with the help of the expression query language XPath. XPath (XML Path Language) is used to navigate and select elements from an XML document [41]. It offers a simple way to define paths and patterns that are used to locate particular nodes or groups of nodes within the XML structure. The hierarchy and relationships between elements are described in XPath expressions, which are made up of a series of steps separated by slashes (/). It enables you to choose nodes according to their position within the document, tag names, attributes, and text content [42].

To some extent, HTML documents can also be used with XPath. Even though HTML is not strictly an XML-based markup language, many contemporary web browsers allow one to query and manipulate HTML documents using JavaScript and XPath expressions. The internal HTML parser of the browser transforms the HTML file into a Document Object Model (DOM) structure that can then be navigated using XPath, thus enabling this functionality [43].

Before starting to create the script and for testing purposes, we used the Postman tool [44] on every HTML call request made to the website. Postman provides developers with a convenient platform to design, test, and document APIs, offering support for various request types such as GET and POST. This tool significantly simplifies the process of API testing and debugging.

3.1.2 Extraction

In order to proceed with the creation of an XPath extraction algorithm in Python, the target website architecture needed to be first investigated. In doing so, we discovered that the website represented all of its web pages as data tables, a good representation to simplify information extraction. Each webpage only had two columns, one for the key and the other for the values. Both column entries could be extracted using the same method and a one-to-one connection can be made between them using a dictionary structure in Python.

The DGSi website has a particular link structure that can be observed in figure 3.2. Its pattern was used to make the fetching dynamic.

The diagram shows a URL with three parts under curly braces: `http://www.dgsi.pt/jstj.nsf/` is labeled "DGSi's Search page URL", `954f0ce6ad9dd8b980256b5f003fa814/` is labeled "Supreme court ID", and `99c60644b825296c802589c900396265` is labeled "Court ruling ID".

Figure 3.2: DGSi's URL Structure

When extracting the court ruling table (Which can be seen on Figure D.1), the fields' names will come in Portuguese and with UTF-8 characters such as ones with tildes. The following fields were refactored from the raw structure to the dictionary from direct correlation from key to value: "Tribunal" (court), "Processo" (process number), "Data do acórdão" (decision date), "Votação" (vote), "Sumário" (summary).

The following fields need to be somewhat changed to facilitate the future parts of the project:

- **Descritores:** Descriptors. This is a collection of keywords that would appear separated with *ENTER* character, so this characters were removed and every descriptor added to an array.
- **Decisão Texto Integral:** Full Text. This is the most important field of the extracted body as it is the one that is going to be used to identify and extract references. For now, during this stage, the only cleaning done was to remove any area that has multiple consecutive *ENTER* characters and replace them with just one character. In a later stage, this *ENTER* character will be helpful to split the text into sentences.

In the end, the document's link was added with the key "Link" to the dictionary. Any other field that was not mentioned was not important to the project but can be easily added if needed.

3.2 Data preparation

After extracting the information available in the documents repository website, each document must be parsed to identify relevant fields. Every field from this raw dataset should be evaluated, optionally modified, and posteriorly added to a JSON schema. The fields mentioned in the

previous section were renamed to be more concise and in line with database nomenclature, as shown in table 3.1.

Table 3.1: Renaming fields from raw dataset to new dataset

Raw Field Name	Changed Field Name	Raw Field Name	Changed Field Name
Processo	<code>_id</code>	Link	<code>link</code>
Tribunal	<code>court</code>	Data do Acórdão	<code>date</code>
Descritores	<code>descriptors</code>	Relator	<code>author</code>
Sumário	<code>summary</code>	Decisão Texto Integral	<code>full_text</code>

The `_id`, `link`, `date`, `summary` and `value` did come as an array with only one value, so they had to be assigned that first value. The `descriptors` and `author` values were already taken care of in the previous iteration, thus it was a direct assignment. To assign a value to `court`, some cleanup was needed namely removing the substring "Acórdão do " as it is irrelevant in this context. A field entry `year` is also added to the schema by converting first the `date` value into a `date` type value using the `datetime` library in Python and then extracting the year-part.

3.3 Database Architecture

As mentioned in chapter 2, MongoDB was the selected technology for the project. This non-relational database offers several advantages that are beneficial to the project, such as scalability and performance, allowing for quick and efficient data retrieval. One of its key features is its flexibility, as it does not require predefined data types for fields unless a schema is explicitly created. However, for consistency, a schema can be created to enforce structure.

One important aspect of MongoDB is that it uses the `"_id"` field as the default primary key for documents within a collection. In the project, the "Processo" field represents the identifier of a court decision document in the Publication scheme, so it was assigned to the `"_id"` field. this allows for efficient retrieval and identification of specific documents in the Publication table collection. More information on the conversion on Table 3.1 and details about the Publication scheme in 3.2.

In order to facilitate the integration of our prepared JSON file into the database, it was necessary to partition it, as it contains information that can be allocated to different tables. The Publication's Table (Table 3.2) encompasses the majority of the publication-related details. On the other hand, the authors' Table (Table 3.4) accommodates author-specific information, featuring a "verified" boolean field that serves to indicate whether the authors' name has been subsequently verified after its addition to the database. Additionally, the `"alias_of"` field allows for the specification of an alternate author name in cases where an alias is present, otherwise, this field defaults to null. The absence of a direct connection between the authors' Table and the Publication's

Table 3.2: Database Architecture - Publication

Attribute name	Type	Example	Insight
_id	String	3696/15.5T8AVR.P2.S1	Publication Identifier, generated from Process Number. "_id" is the default ID nomenclature for identifiers in MongoDB
link	String	https://www.dgsi.pt/...	Link to the official repository of the publication.
descriptors	String Array	["Posse", "Arrolamento"]	Keywords that are used to better describe the publication or information presented. They simplify the search for a publication.
type	String	acordao	Type of document
date	String	12/12/2012	Date of the publication
year	Integer	2012	Year of the publication. Made to be easy as it is more important than date.
court	String	Tribunal da Relação do Porto	Court, if applicable.
summary	String	"I - O STJ conhece de matéria de facto[...]"	summary of the publication, in case of necessity from a third party.
full_text	String	"I - RELATÓRIO Em dissensão com o julgado proferido[...]"	Full text of the publication used through the project as it is where the algorithm looks for the references and extracts those.

Table stems from the existence of a many-to-many relationship, where a publication can have multiple authors, and an author can be associated with multiple publications. Consequently, a junction table, identified as Table Publication-author (Table 3.5), was introduced to manage this relationship. Lastly, the Reference Table (Table 3.3) was established to store information pertaining to references, including the connection to both the referencing and the referenced publication. This design decision to maintain a detachment between the authors' Table and the Publication's Table is primarily motivated by the inclusion of comprehensive information, with the exclusion of direct links, to ensure the preservation of data integrity. Such an approach enables seamless implementation of future projects, allowing developers to present references with consistent terminology and author aliases. The overall database schema and the interrelationships among the tables are visually represented in Figure 3.3.

Table 3.3: Database Architecture - Reference

Attribute name	Type	Example	Insight
_id	String	63eecb863d7c3d0[...]	Reference Identifier.
name	String or Array of Strings	Fábio Nogueira	Name(s) of the author(s)
pages	Array of Strings	['64','65','76','+']	Pages in reference. It was decided to use strings to use the '+' sign and to avoid creating complexity on that boolean option.
volumes	Array of Strings	['3','4']	Volumes in Reference. It can also use '+' sign. Volumes normally appear as Cardinal Numbers or Roman Numbers. Optional.
notes	Array of Strings	['1a', '2b']	Notes in Reference. It can also use '+' sign. Optional.
publication_caller	String	3696/15.5T8AVR.P1	ID of the publication where the citation lies on.
publication_receiver	String	3696/15.5T8AVR.P2	ID of the publication that the citation references. This can be set as null if no publication is found. In-depth explanation on Chapter Extraction
editor	String	"Nome De Editor"	Editor's name. Optional.
edition	String	"2ª Edição"	Edition's information. Optional.
year	Integer	2020	Year. Optional.
other	Array of Strings	["Rlj" "Other Name"]	This can be any information that was not caught by the algorithm but is inside its limits. Parts such as courthouses, other names for the documents or full dates will lay on this field. Optional.

Table 3.4: Database Architecture - Author

Attribute name	Type	Example	Insight
_id	ObjectId	63eeeb863d7c3d0[...]	Author ID automatically generated.
name	String	Fábio Nogueira	authors' name
alias_of	ObjectId	63eeeb863d7c3d0[...]	If the author is an alias of another author, the main authors' id should be inserted in here
verified	boolean	false	If the author name is verified by a system administrator, it can be put as "true". If the algorithm extracts a name that is not on the list, it adds it with this boolean as "false"

Table 3.5: Database Architecture - Publication-Author

Attribute name	Type	Example	Insight
_id	ObjectId	63eeeb863d7c3d0[...]	Publication - Author ID automatically generated.
id_publication	String	3696/15.5T8AVR.P2.S1	Publication Identifier of Table 3.2.
id_author	ObjectId	63eeeb863d7c3d0[...]	Reference Identifier of Table 3.4.
author_type	String	Relator	Type of Author in publication.
author_order	Integer	3	Value that represents the authors' position in the publication's authors' list.

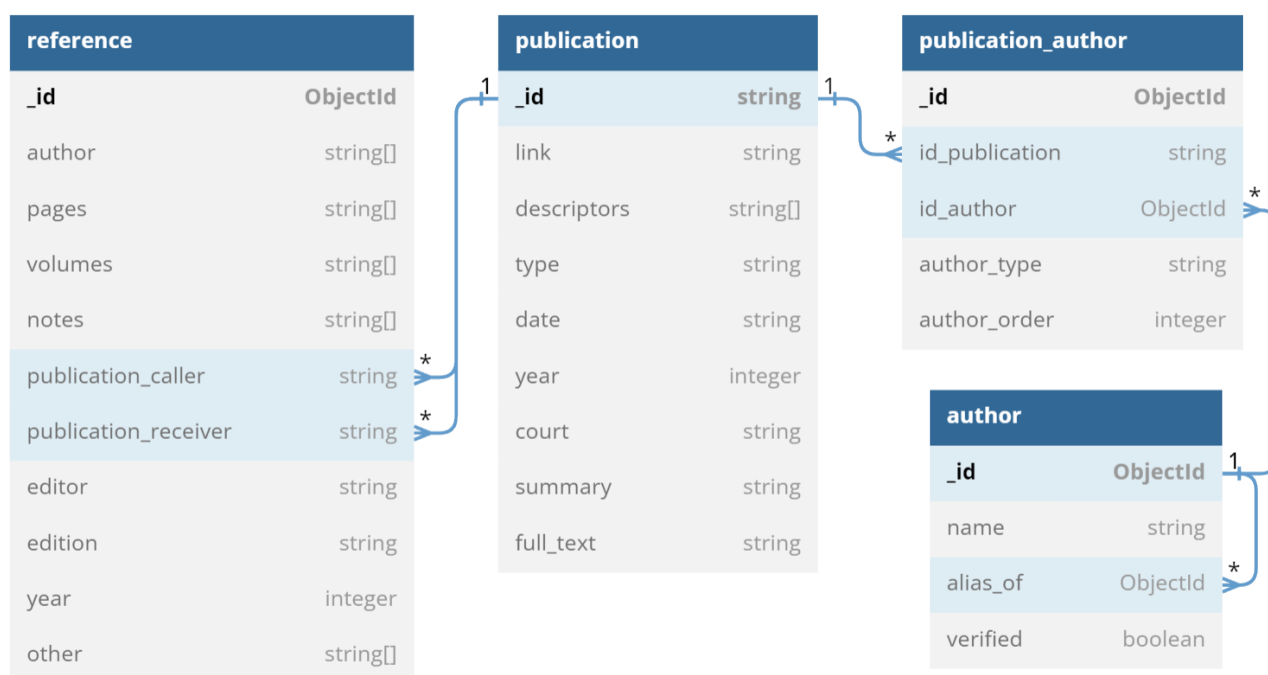


Figure 3.3: Database schema involving all tables and relations.

3.4 Data Preprocessing

The previous data cleaning is sufficient to store the relevant data on a court decision document in a format that enables its use in the next stages of the project. However, some further data preparation and tokenization are still necessary.

3.4.1 Data Preparation and Tokenization

When preparing data to tokenize information from the full text, it is important that this behaves as the models expect it to. To achieve this, there are some changes to the actual text that are necessary to be made. The following cleaning remove or replace undesired or abnormal characters:

- A newline character is replaced with a space character;
- Any weird character, such as non-breaking space, is removed; Sometimes a non-breaking space character (NBSP) appeared and had to be removed;
- Characters that are very similar, like hyphen(-) and dash (–) are perceived as different by the compiler, so they are harmonized to dash, as it is the character used in Portuguese keyboard's layout;
- The dash is scattered in many places, thus if mistakenly used can generate tokenizing problems. Due to this, the following changes were made: ". -" to "- " and ".- " to "- ";
- Parenthesis can introduce problems to the tokenizer due to its use together with the dot character as a way to present new information. In this case, the following changes were made: "). " to ") " and ". []" to "[].";
- When a dot appears before a quotation mark, the program removes the dot;
- Successions of characters should be edited or shortened. This is important on phrase chunks like ". " to "." and "; " to ";" because the tokenizing tool is sensitive to those spaces to divide phrases in the full text;
- In some particular cases, there were singular, uppercased letters followed by two dots. In this case, one of the dots was removed;
- Finally, DGSI, when there are documents with a reference list below, uses multiple underscores (__) to define the end of the document. In this case, we replaced this line by a tag referencing the separation, as it should not be evaluated by the algorithm.

References made by brackets in documents that are already used need to be normalized. In this case the following changes to the text were made when:

- A phrase ends with the bracketed reference made after the punctuation and followed by a newline character, the bracketed reference is moved to before the punctuation;
- A phrase ends with the bracketed reference made after the punctuation and followed by words afterwards, the bracketed reference is moved to before the punctuation;
- There are multiple references made in a row, those are grouped together inside the same brackets and divided by a slash.

3.4.2 Data Preparation to Sentence Divider

Once the full text has been cleaned and prepared, the next step is to divide it into sentences. We use the Python Natural Language Toolkit ([NLTK](#)) library to accomplish this important task. As mentioned before, [NLTK](#) is quite an effective tool for processing human language and provides us with a wide range of functionalities, namely the capacity to segment and divide text into useful phrases or sentences.

The phrase division capabilities of [NLTK](#) can be used to split the text into coherent and contextually meaningful chunks. Punctuation marks, like periods, question marks, and exclamation points, which frequently denote the end of a sentence, can serve as the basis for these divisions. As an alternative, [NLTK](#) can make use of more sophisticated methods like natural language parsing to pinpoint the boundaries of sentences based on grammatical conventions and syntactic structures.

[NLTK](#) has a class called `PunktParameters` [[45](#)] to deal with the use of abbreviations' punctuation when splitting the text and thus avoiding splitting errors. This class is used to customize the behaviour of the `Punkt` sentence tokenizer and uses unsupervised learning to determine the sentence boundaries based on multiple features (more details are given in figure [3.4](#)). It allows the developer to change parameters like abbreviations, tokenization preferences, and training data selection. The final version only uses its capabilities to introduce some Portuguese abbreviations gathered from an online dictionary forum [[46](#)].

To ensure flexibility and easy modification, the abbreviations utilized in the text processing pipeline are stored in a dedicated configuration file. Any updates or modifications to the abbreviations can be made without affecting the main codebase by saving the abbreviations separately from the code. This strategy improves maintainability and makes it easier to modify the system to meet the developer's requirements.

3.4.3 Authors' Data

At this stage, the Publication Table has been populated with the hand-picked documents selected by the team. Now, the authors' database table needs to be filled in. To accomplish this, the team utilized the FDUP's (Faculdade de Direito da Universidade do Porto) researcher's open

On completing this process, the list of names underwent a minor cleaning procedure to remove entries that contained numbers (e.g., "a.C.", as in Ante Christum for dates, or names with identifying numbers) or names that were deemed too short, as they could potentially have wrong name matches. Such entries were deleted from the list. This cleaning process was carried out for all authors in order to populate the authors' Table.

3.5 Summary

This chapter introduced the steps taken to create the dataset to be utilized for the research, along with the data-cleaning procedures involved. It highlights the importance of dealing with some peculiarities in the data, thus emphasizing the need for careful handling of potential complexities. The tools employed for data cleaning help to achieve better accuracy and reliability of the dataset. Additionally, the techniques and methods applied to preprocessing the data elements present in a document were also introduced. This stage, where we extract court rulings from the DGSJ's website, parsed and inserted them in the database for later use, provides a solid foundation for the subsequent stages of the research, enabling a robust and rigorous examination of the data.

Chapter 4

Identification of references to legal literature

This chapter focuses on the development of an algorithm to be used for the identification of references (or citations) to legal literature included within the preprocessed court decision documents. The algorithm primarily uses information about stored publications, both the full texts of court decisions and metadata characterizing a legal publication (book, article, etc.), as described in table 3.2, and also needs to access the list of known authors, as described in table 3.4, in order to recognize the authors being cited in court decision documents.

This stage relates to the second part of our project’s pipeline, as seen in figure 4.1. In this chapter, we already have the data related to the publications inside the database so the next step is to parse them to identify citations and then output a clean, parsed document ready for the next stage, metadata extraction.

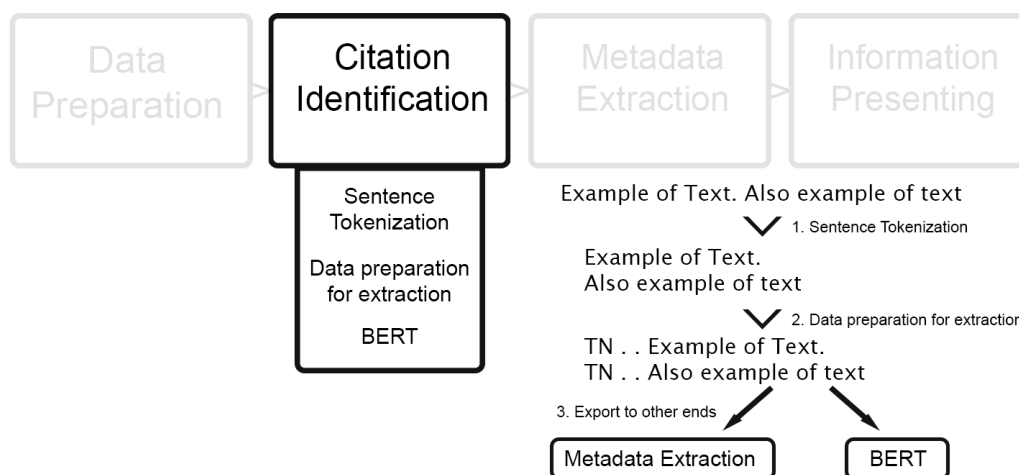


Figure 4.1: Project’s Pipeline: Citation Identification Stage

4.1 Sentence Tokenization

With the data now cleaned and the *punkt* parameters set for sentence tokenization, the prepared data is ready for further processing. Various approaches were explored for sentence tokenization and quote identification, and this section only highlights the most noteworthy employed in the project.

CiteSeer stated that there were three types of citations [36]:

- Those that are easy-to-identify as they display a clear, unambiguous and normalized citation syntax;
- Those that are hard-to-identify as they may appear with different formats, without a proper systematization, but can be understood by a machine;
- And those nearly impossible-to-identify that they are a citation, even for a person because it is not a direct reference, only has a number between brackets referencing something in the references list. This last one is avoided by the algorithm as its uncertainty should not be used counted towards validation.

A crucial aspect of this research project involves developing a systematic framework to accurately perform the identification of citations within the study's context. While these methodologies enhance accuracy, caution must be exercised due to the possibility of false positives. The next phase involves retrieving known authors' names from the database, ensuring comprehensive evaluation and improving identification accuracy. This strengthens the authenticity of the research outcomes by aligning the identified references with the relevant project documents.

From a technical standpoint, it was decided to gather all authors' names from the database and compile them into a list before running the algorithm. Depending on the implementation, this method enables the algorithm to either iterate through each name or run a search query for each potential reference. The limitation of storing the list of names in system memory rather than utilizing the vast storage capacity of a database can present challenges when it comes to scalability. Alternative project-planning strategies are reviewed in Chapter 7 in an effort to mitigate potential problems and more effectively address scalability concerns. These alternatives seek to streamline the procedure and remove any potential memory-related restrictions, improving scalability.

The procedure to do this was to first identify if a pattern existed (an exemplary table can be found on Table A.2) and then verify, for all names, if the full name is present. Identifying the pattern before looking for a name made the code faster by 71 seconds, which is 92.21% (Figure 4.2 and Figure B.1) faster than starting by the name and had the same results comparing to the other way around. This happens because the pattern search that the library "re" uses is faster than looking into an array of names inside the sentence array. This means that the name is only searched if a pattern is located.

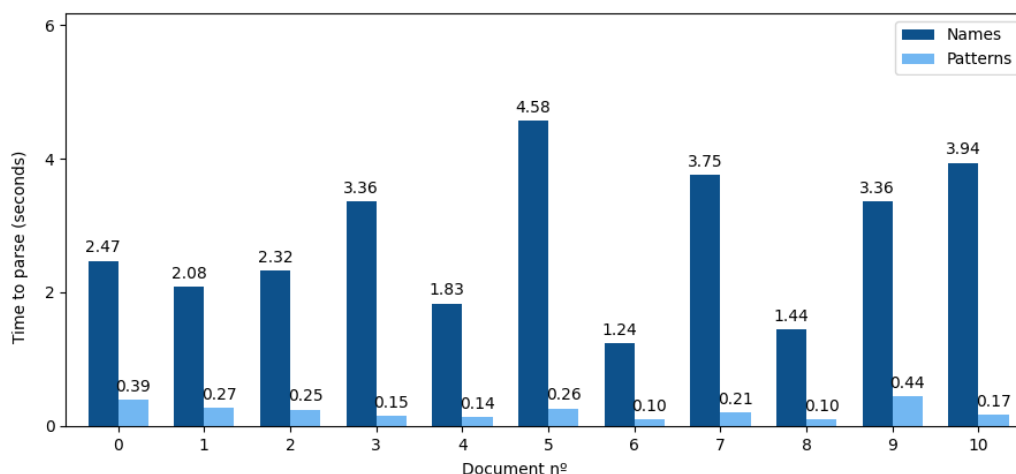


Figure 4.2: Graph plot relating identification via patterns first and names first.

The pattern gathering was a job conducted by the team, consisting in the F. Nogueira, F. Silva, M. R. Guimarães and S. Bugla, that focused on finding more unique ways of referencing a document. Examples of patterns include "como ensina" (meaning "as taught"), "em sentido próximo [...] se pronuncia" (where "[...]" signifies any word or characters), and "na doutrina" (meaning "in the doctrine"). Although it may not always be the case, these patterns act as markers of prospective citations in a phrase's syntax. In other cases, citations may appear in between parentheses, after the conclusion of a phrase, or by using unseen-before citation techniques.

The algorithm must include additional keywords that do not rely only on phrase structure in order to handle these situations. Keywords such as "pags." (short for "pages"), "cfr." (abbreviation for "conforme," meaning "refer to"), "in," or "vols." (short for "volumes") might appear within the citation. The system can now find references regardless of their location or unusual usage by considering these extra keywords.

As previously stated a decision was made on the algorithm to identify references by name. This decision had different iterations as various approaches were made. As a first stage, we had a simple researchers list, which lacked variability of different names and thus was not used. But, it helped in deciding what alternatively one should do in the project. After this, an attempt to gather various first names and possible last/middle names was made. This attempt was fast as it used two lists that would be used for many different combinations of names but had many flaws.

First, Portuguese names sometimes have prepositions. This would be considered, in a first iteration as part of the name, so it had to be mitigated by including an optional word in the middle of the name. If this was not mitigated, strings such as "morte das" (as in "death of") would be considered a name, just because "morte" can be a name and "das" also appears in names, but it would not be a real name.

Sometimes editors use last names before first names, which would cause some issues as the

algorithm would first search for any number of first names followed by any number of last names (having prepositions as options in between). A couple of rules were made because of this: first and last names should not jump in between, as last names could only be followed by first names and followed by last names in the worst-case scenario. If the name started with a group of first names it should only be followed by a group of last names, and this ends the name. As the reader can easily understand from this, a grammar rule can be done for this, so it was done as follows and will be explained in more detail in the next chapter 5:

```
NAME: { <NPROP2> ( (<NPROP> | <PREP>) ? <NPROP2> ) + }
```

Grammar rules were not used as the default method for identification in the project's final iteration. To ensure consistency and to avoid using multiple different names or aliases, it was decided to give priority to researchers who had been confirmed by the team. The project prioritized the established identities rather than depending solely on grammar rules in order to maintain a greater level of accuracy and reliability in the identification process. This thesis will work around grammar rules in the next chapter 5 as it was more prominently done in that phase.

As a final stretch, as it was said, the final iteration of the code only uses the list of known authors. At first, this introduced some difficulties because of the way it was implemented. Since names can be rearranged in multiple forms, it was important to avoid having the names strictly the way they were in the database. For that reason, it was used a python method that checks if all elements of an array are inside of another array, independently of its order. To make this efficient, it is important to divide the sentence that is being evaluated into an array of words. To do this the program first deletes any instances of punctuation, special characters or numbers and only then it divides by spaces. It is also important to have special attention to characters with accents, as they are not in the default encoding, so a UTF-8 encoding support needs to be added to the project as we are using Portuguese words.

Chapter 6 explores potential steps that a future developer can follow to make this identification process even faster, using the same bases.

4.2 Data preparation for extraction

Now that the program is ready to identify, it will try to identify the references, sentence by sentence in the full text of the document. When it finds a pattern it will look through all names and check it against the sentence that has the pattern. When a citation match is found, it then prepares the data for the next phase, citation or reference extraction.

4.2.1 Output of identified references

To facilitate the evaluation process, the program will create lines with information for each phrase being evaluated. Each sentence (a line of characters) will incorporate the addition of the keywords "TP?" or "TN?" (representing True Positive or True Negative) at the beginning. However, it is important to understand that the determination of a sentence as a True Positive or True Negative is beyond the scope of the algorithm itself, this identification is only to help human evaluation. This classification relies on human judgment in order to establish ground truth data. The algorithm cannot make definitive decisions in this regard without having ground truth data from which it could learn, hence the inclusion of the character "?" serves a dual purpose. Not only does it indicate that the sentence has not been accurately classified as "TP" or "TN," but it also aids the human reviewer in keeping track of the sentences that require manual verification.

For sentences that were identified with "TP", two helper columns were added. One indicates the name of the author that was identified and the other the pattern identified. This is only applicable to identification algorithms that used names and patterns, as other algorithms such as the learning method or grammar rule identification did not have this distinction.

For sentences that contain numbers inside square brackets, a value of "-" is added instead of TP or TN, followed by "bra", as in "bracket", because, as said previously, this type of sentence should be ignored as it can lead the algorithm to wrong assumptions. These sentences should also be avoided when training a learning algorithm. Here is an example of this:

```
TN not . . Na contestação, que apenas os réus AA, BB e "CC-Promoção
    Imobiliária Unipessoal, Lda."
TP ref |Maria Raquel Guimarães| | pp\ . | Expressões como "dinheiro
    de plástico", "moeda electrónica", "telemática", "caixas
    automáticas", "porta-moedas electrónico" são exemplos de termos
    que passaram a fazer parte do léxico corrente dos dias de hoje
    ( Com mais desenvolvimento, cfr. Maria Raquel Guimarães, "As
    Transferências Electrónicas de Fundos e os Cartões de Débito",
    Almedina, 1999, pp. 11 e 12.).
-bra . . Além do pai, os outros familiares, que, segundo o critério
    proposto para o caso de morte, teriam direito pessoal de
    satisfação, parece deverem tê-lo aqui também [1].
```

Furthermore, to enhance the comprehensibility of the program's output during the manual verification process, two additional components are included in the final string. First, the identified pattern is appended, providing insight into what the program has recognized. Second, the corresponding name identified in the phrase is also included. By providing these additional details, it becomes easier for the manual verification team to interpret the program's findings, whether they are correct or incorrect.

Tokens	Token IDs	Attention Mask
[CLS]	101	1
exemplo	1416	1
de	125	1
texto	4054	1
.	119	1
[SEP]	102	1
[PAD]	0	0
...	0	0
[PAD]	0	0

Table 4.1: Example text and its transformation into tokens and attention mask

Once the verification phase is completed, the string array, comprising all the accumulated information with each sentence labelled as TP or TN, is saved to a file. This file serves as a basis for subsequent manual verification by the team. During this review, each line is carefully examined to determine its classification as a True or False Positive, as well as a True or False Negative.

This new file will have two purposes: it will be used to confirm if the full text is already divided into sentences and if each sentence includes a possible reference or not, for reference extraction, and also will be useful to train the BERT AI algorithm.

4.2.2 BERT

As mentioned earlier, all sentences that compose the text of a court decision were manually verified. The main objective of this tiresome task was to ensure a quality dataset for BERT. The success of BERT's identification of true references highly depends on having enough ground truth data from which it can learn.

In the first stage, it is necessary to remove all the information between the first part of each string (where TP, FP, TN and FN lie), and the actual sentence. The code will grab all negatives and put them as 0 and grab all the positives and put them as 1. This data was then inserted in data frame methods from the Panda library. It also used the transformers library because it has methods specifically for BERT, such as *BertTokenizer* and *BertForSequenceClassification*, which is the type of classification that we are exploring. *BertTokenizer* is used to create the tokenizer from a pre-trained model. We used a Portuguese model from the neuralmind team, BERTimbau [47] [48]. Table 4.1 presents a random sentence where each word-level split has a Token ID.

BERT requires the following preprocessing steps:

- Add special tokens: [CLS] at the beginning of each sentence with id 101 and [SEP] at the

end with id 102

- Make sentences all of the same length, by adding a padding to a max value tokens (512 [20]). Longer sentences are truncated. [PAD] has an id of 0.
- Create an attention mask and a boolean list is generated to indicate whether the model should consider the referred token when learning its contextual representation. The token "[PAD]" will be assigned a value of 0 in the mask

The process can be visualized in Figure 4.3. Using the tokenizer method *encode_plus*, it returns a list of token IDs, a list of token type IDs, and an attention mask list.

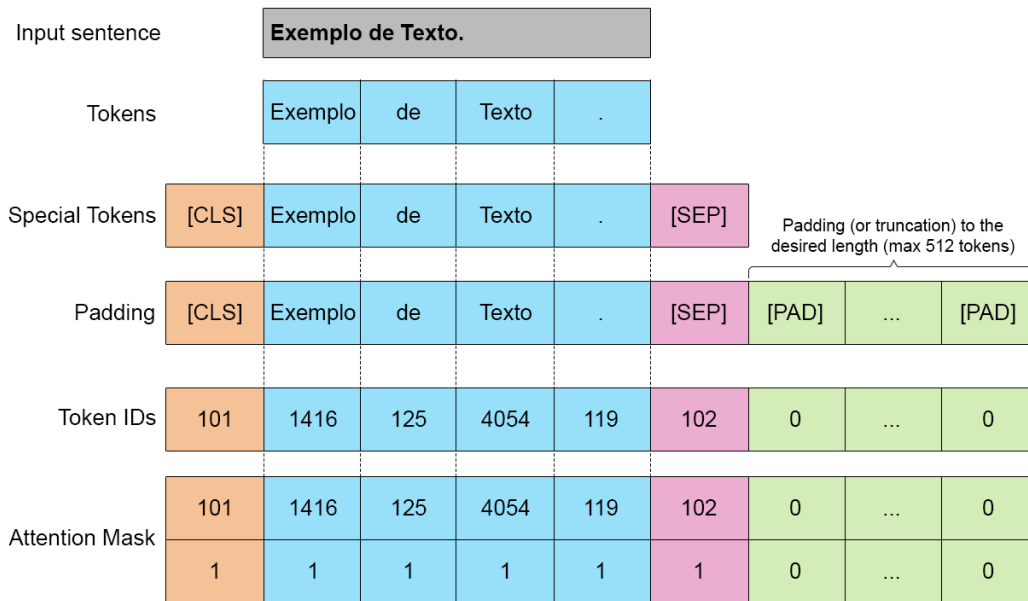


Figure 4.3: Tokenization process pipeline visualizer

In Figure 4.4 can be observed the token IDs array for another given text sample, inside a tensor wrapper object, and recognize the presence of special tokens, [CLS] and [SEP] and padding to the desired max length.

```
tensor([ 101, 1048, 2595, 1012, 1010, 2139, 1018, 1012, 1016, 1012,
        2639, 1010, 1039, 1007, 1046, 1012, 2639, 1010, 1015, 1010,
        10114, 1090, 1012, 102, 0, 0, 0, 0, 0, 0,
        0, 0])
```

■ [CLS]
 ■ Token IDs
 ■ [SEP]
 ■ [PAD]

Figure 4.4: Token IDs' array as a tensor object and their respective class

The data is then split into an 80/20 model, where 80 is equivalent to 80% of the dataset being used for training and 20 corresponds to 20% of the dataset used for validation. To accomplish this split and training, the algorithm employs *pytorch*'s methods, namely *TensorDataset* and

train_test_split. Then, these two datasets are wrapped into a *DataLoader* to prepare them for training (Code sample C.2 [49]).

Based on experience reported in the BERT paper [20], the optimal parameter values are found in between these values:

- Batch size: 16 or 32;
- Learning rate (Adam): 5e-5, 3e-5, or 2e-5;
- Number of epochs: 2, 3, or 4.

To correctly identify the positive impact of this approach, we need to calculate validation metrics of the training process. To do so, we introduced methods to calculate accuracy, precision, recall and specificity (Code sample C.1 [49]). Figure 4.5 presents such calculations. The obtained results and the parameters used will be further discussed in chapter 6.

- True Positive (TP): correct prediction of actual class 1;
- False Positive (FP): wrong prediction of actual class 0;
- True Negative (TN): correct prediction of actual class 0;
- False Negative (FN): wrong prediction of actual class 1;
- N (Total number of predictions): TP + FP + TN + FN.

$$Accuracy = \frac{TP + TN}{N} \qquad Precision = \frac{TP}{TP + FP}$$

$$Recall = \frac{TP}{TP + FN} \qquad Specificity = \frac{TN}{TN + FP}$$

Figure 4.5: Validation calculations to process training's accuracy, precision, recall and specificity.

The final step of this approach is the prediction. At this stage, we already have a model ready and trained to predict if a given sentence has, or does not have, a reference to a legal document. As with previous sentences, the one to be evaluated must pass through the same preprocessing algorithms, which includes IDs extraction and Attention Mask extraction. Using these values, the model processes the data and generates a boolean prediction.

4.3 Summary

This chapter described the method used to discover citations to Portuguese legal literature and how that can be achieved with a low computational time. It also described the application of transformer architecture **AI!** (**AI!**) technologies, as a possible better alternative approach for the identification task, provided there are quality datasets available.

In this chapter we were also able to answer the first question introduced in chapter 1, "*How to create an algorithm that selects citations, or quotes, in a Portuguese text through collected superior court decision documents with a diversity of citation syntax?*".

Chapter 5

References Extraction

This is the last phase of the project pipeline as shown in figure 5.1. This part will use the last chapter's output file (Chapter 4) to extract useful data from the identified references. Initially, this phase was seen as out of scope considering the project's premise but was added due to its interesting procedure and useful result.

This stage starts with the information parsed in the last chapter and processes it via grammar chunking to simplify the identification and extraction of the sentence parts (or metadata) that constitute the citation. We must prepare this metadata for future works on data visualization and presentation.

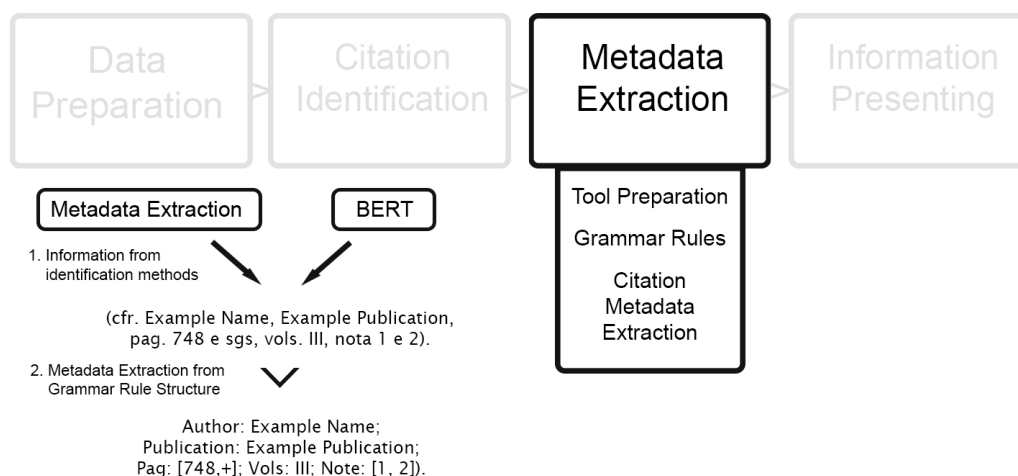


Figure 5.1: Project's Pipeline: Metadata Extraction Stage

5.1 Extraction Tool Preparation

Extracting information from citations can be challenging due to the various patterns and formats in which author names are written in superior court documents. However, one approach to

mitigate this problem is by leveraging a feature in the Natural Language Toolkit ([NLTK](#)), called Part-of-Speech ([PoS](#)) tagging. [PoS](#) tagging involves assigning lexical categories or tags to known words in a sentence. These tags represent the grammatical roles and categories of the words in a sentence.

NLTK provides a collection of word and tag tuples that form a tagset. This tagset is utilized to assign appropriate tags to words in a citation. By applying POS tagging to the citation text, it becomes possible to identify specific components such as author names, publication pages, publication dates, and other relevant information.

The use of NLTK's POS tagging feature can enhance the accuracy and efficiency of information extraction from citations. By utilizing the tagset and analyzing the lexical categories of words, researchers can develop rules or patterns to identify the desired information more effectively, regardless of the variations in citation formats used by different authors.

A POS-tagger processes a sequence of words and attaches a part of the speech tag to each word. As it can be seen in the following example, the sentence is divided into word chunks and then tags are added to each word.

```
Text: "And now for something completely different"
[('And', 'CC'), ('now', 'RB'), ('for', 'IN'), ('something', 'NN'),
 ('completely', 'RB'), ('different', 'JJ')]
```

Lexical ambiguity arises from the fact that many words can possess multiple meanings depending on their context. For example, the word "refuse" can function as both a verb denoting "denial" and a noun referring to discarded or worthless material. In linguistic analysis, these homonymous words are assigned to distinct lexical categories, allowing for accurate representation and interpretation.

The utilization of different tag tokens within the lexical field varies across languages. For the project, a Portuguese dictionary must be used. In NLTK terms, this is called a corpora. Each corpus, or corpus, will have its own set of words and designated lexical category. Specifically, the official Brazilian Portuguese corpus is known as "Mac-Morpho" [\[50\]](#) was employed, as it provides comprehensive documentation outlining each lexical category's usage and contextual nuances [\[51\]](#). To facilitate understanding and analysis, a detailed table showcasing the available tags within the Mac-Morpho corpus is provided in [Table A.3](#).

Despite the extensive word coverage of the corpus, there are instances where certain words are categorized as "unknown". This issue often arises with abbreviations, even though the corpus is designed to handle them. To address the problem of significant words being classified as unknown, a solution involves adding custom words to the corpus. For this project, a custom list was created with each line representing a word-tag pair and incorporated into the code to augment the corpus's vocabulary and improve accuracy (Examples list on [Table A.4](#)).

After enhancing the lexical dictionary, it became necessary to modify certain tag assignments.

In this case, we needed to override certain tags for names and punctuation. The decision to overwrite names is driven by the need to align the names identified by NLTK with our specific authors' list, ensuring that names are accurately identified. Additionally, it is important to distinguish between proper names and other uppercase words. For instance, the chunk "Supremo Tribunal" refers to the supreme court rather than a specific name, and should not be tagged as "NPROP" (the mac-morpho tag for proper names). To address this, the authors' list extracted from the database is parsed, generating a comprehensive list of possible names. Connectors such as "de" or "e" are removed, and the names are validated to ensure they appear in uppercase within the phrase. This ensures that all names are properly identified.

Furthermore, custom tags are added for punctuation to facilitate the parsing of citations and identification of citation boundaries. This will be further discussed during the phase of parsing extracted citations. The following example illustrates a parsed text that incorporates the modifications mentioned, utilizing the mac-morpho corpus and the aforementioned changes.

The word "in" was also inserted with the tag "IN" as it is a recurrent (but not certain) keyword that is used in dynamics citations. Dynamic citations are citations that are very different from a somewhat "normal" citation. For the project we considered a citation to be normal when it starts to be composed by a name, followed by the title, then the pages, then volumes, year, notes, editor and edition, without particular order in the last ones.

Text: "Exemplo de Texto (Texto que antecede citação, cfr. Maria Raquel Guimarães, \"As Transferências de Débito\", Almedina, 1999, pp. 11 e 12.)."

```
[('exemplo', 'PDEN'), ('de', 'PREP'), ('texto', 'N'), ('(', 'PUNC'),
('texto', 'N'), ('que', 'PRO-KS-REL'), ('antecede', 'V'),
'citação', 'N'), (',', 'PUNC'), ('cfr', 'N'), ('.', 'PONT'),
('maria', 'NPROP2'), ('raquel', 'NPROP2'), ('guimarães', 'NPROP2'),
(',', 'PUNC'), (''', 'PUNC'), ('as', 'ART'), ('transferências', 'N'),
('de', 'PREP'), ('débito', 'N'), (''', 'PUNC'), (',', 'PUNC'),
('almedina', 'NPROP2'), (',', 'PUNC'), ('1999', 'NUM'), (',', 'PUNC'),
('pp', 'PAG'), ('.', 'PONT'), ('11', 'NUM'), ('e', 'KC'),
('12', 'NUM'), ('.', 'PONT'), (')', 'SEP'), ('.', 'PONT')]
```

5.2 Grammar structure for chunk identification

The next phase of the information extraction pipeline is entity detection. In this step, we use grammar rules to identify citations within the sentences that are being evaluated.

To better create those grammar rules we first need to understand what chunking means on NLTK. Chunking is the ability to unite one or more entities into a multi-token sequence. As each entity

has its own POS tag, a group of these will be a higher-level chunk. As we can see in figure 5.2, an example of a text in an English corpus, the wider the box wrapping other boxes, the higher will the chunk level.

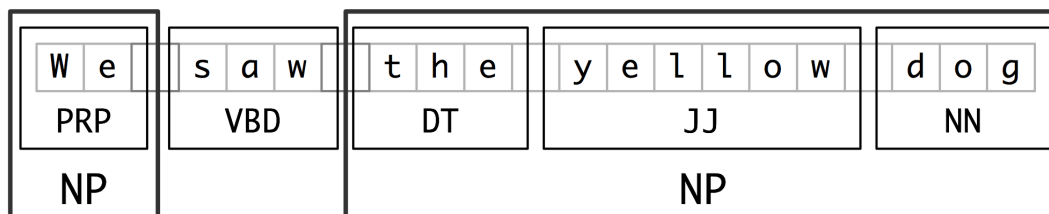


Figure 5.2: Segmentation and Labeling at both the Token and Chunk Levels

Chunk grammar uses tag patterns to describe sequences of tagged words. This tag pattern is a sequence of POS tags delimited using angle brackets, for example, `<N>?<PREP>*<NUM>`. More information on this can be found in NLTK's manual but for our project, this was what was needed from chunking.

The grammar rules are chunks organized with regular expressions to identify some part of the sentence. For names, we created a rule that states that a name should start with a name from our verified authors' list, then it can be followed by one or more common names and a verified name, a preposition and a verified name, or just a verified name. The rule would be:

```
NAME: {<NPROP2> ( (<NPROP> | <PREP>) ?<NPROP2> ) + }
```

The page numbers, volumes and notes, all follow the same logic. The word stating if it is in fact a page, volume or note, followed by, optionally, a dot, followed by a number and then, optionally, more numbers separated by punctuation or article. The first part until the dot was made into a smaller chunk to better organize the ruling system. It is important to note that volumes can be displayed as Roman numbers so a cleaning step should be done to include this as a possible number or volume. In this project, we decided to make Roman numerals as numbers, with the tag NUM. Pages and notes can end in a string that states "and following" as in "and next pages as well". There was no indication that volumes could have such wording. The following rules were added:

```
PAG_DOT: {<PAG><PONT>?}
```

```
VOL_DOT: {<VOL><PONT>?}
```

```
NOT_DOT: {<NOT><PONT>?}
```

```
PAGS: {<PAG_DOT><NUM> ( (<PUNC> | <KC> | <ART>) <NUM> ) * (<KC><SGS>) ?}
```

```
VOLS: {<VOL_DOT><NUM> ( (<PUNC> | <KC> | <ART>) <NUM> ) *}
```

```
NOTS: {<NOT_DOT><NUM> ( (<PUNC> | <KC> | <ART>) <NUM> ) * (<KC><SGS>) ?}
```

These three rules could be merged as 2 single rules, but for a future step where they are analysed, it is important that this distinction is made.

We could not see a good way for identifying the name of the publication, the editor or the edition, as these are composed of normal words and normal phrases. In a future step, we will consider more options to correctly extract these.

Now that we have all rules that are possible to extract the metadata, we can create a rule for the full reference. This rule should start with the name, followed by any length of characters and ending with pages, followed by, optionally, volumes and notes, without particular order. It has to be prepared to have multiple names and sometimes with a dynamic citation that includes the word "in". The final rule is the following:

```
(<NOME><.*>+? (<PAGS> (<PUNC><NOTS>) ?) (<SEP> | <PONT> | <PUNC> | <KC>)) |
(<NOME><.*>{, 2} ?<IN><.*>+? (<SEP> | <PONT> | <PAGS>) <KC>?)
```

As shown in figure 5.3, there were essentially two routes that could be pursued. One for a "normal" citation and the other for a more "dynamic" citation. Both start with the name and are followed by the minimum number of words possible until it reaches the pages or the end. This is achieved by using ">+?" in the regex expression. This solution, contrary to ">+" which is greedy, is a reluctant [52], and will search for the shorter route. This is important because if we have two citations in the same sentence, the greedy algorithm would identify everything from the first citation name until the end of the second one.

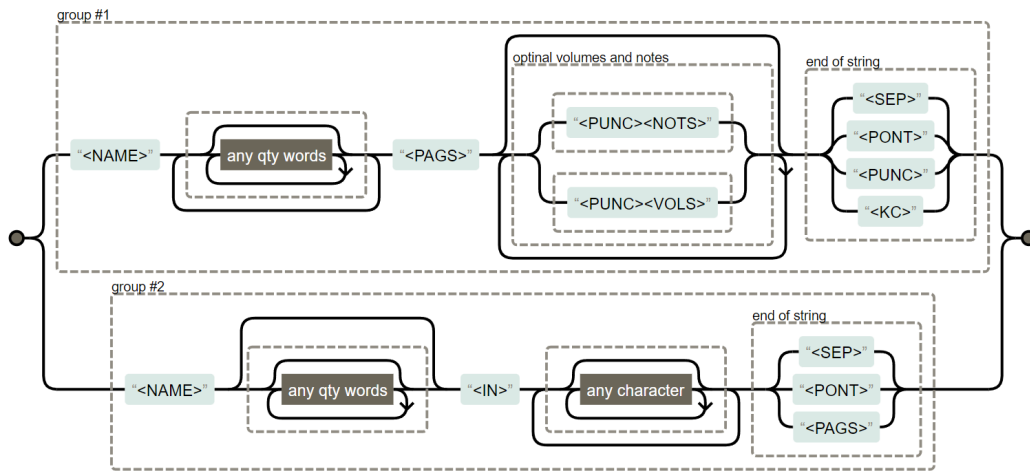


Figure 5.3: Segmentation and Labeling at both the Token and Chunk Levels

Figure 5.4 presents a full representation of a simple citation and its grammar tree. This tree has a root, represented by the letter "S", but as in the given example, the only child branch consists of the reference, represented by "REF". This REF then had multiple children that each represents a subtree or a leaf pair word tag.

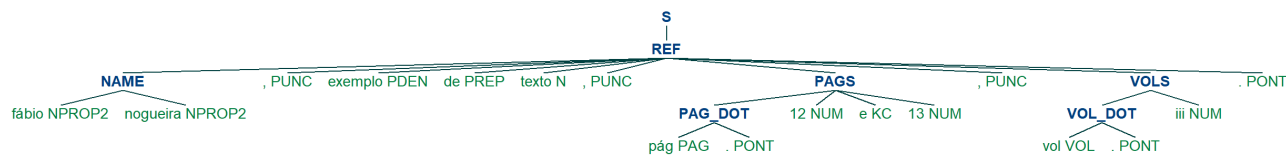


Figure 5.4: Citation's Chunk Tree Example from Python's draw tool

5.3 Citation information extraction

Now that every chunk is correctly identified, we can start to extract the information from the citation. Every part of the phrase has its own caveat so it's important to structure it without generalization. A single sentence can pass n times through this process as it can contain multiple citations.

The information extracted from each tag that can be selected from the rules is the following:

- **NAME:** The program will check every branch of this subtree to create the name. When the name is acquired, the program will check the database for an entry of the name and, if found, it is added, with a relative keyword, to an array that represents the full citation's metadata.
- **PAGS and NOTS:** Pages and notes behave in the same way but have to be treated as separate so the keywords are different. For each value of the branch, we verify if it is a number or a "follow" keyword. If it is a number, it is added to an array, if it is a "follow" keyword, such as "e seguintes" or "sgs", it is added a '+' to the array. In the end, this array is added to the main citation array.
- **VOLS:** Volumes behave in the same way as the previous pages and notes but they can have Roman numerals, so for that particular reason they must have a unique treatment.

For those chunks that could not be identified via grammar rules, personalized treatment must be done. For the subtree being identified, every branch is checked whether it is a subtree or an element. If it is a subtree and its tag is not one of the tags created in the grammar rules, the program will open the same method inside of it in a DFS style. If it is an element, it checks if it is not a termination tag, such as PUNC or SEP. If it is not, the element's word is added to an array. When the word has indeed a termination tag (and the array is not empty), the program will behave in the following way:

- If it is the first time reaching a termination tag, it means that the words gathered are part of the title. This is because the title of the referred work comes after the name of the author or after the keyword IN. Then the array with the words is merged into a string and added to the final citation array with all the citation parameters;

- If the number of words is less or equal to 5, and one of the words is a digit between 1000 and 3000, we consider it as a year and add that keyword to the citation array. The decision for considering just 5 words resulted from our observation that the large majority of the times we found a citation with just the date exposed, but in very rare ones we found a string with day and month, so there is a need to include, at most, a string such as "20 de abril de 2023". If there is a date that we can not find, there is no problem as we will see afterwards;
- If there is a word "editor" or "editora" in the words gathered, (masculine and feminine types of "editor" in Portuguese) we can assume that the string is related to the editor, so it is added with the keyword "editor";
- Like the previous item, the same happens for "edição" (meaning edition), but in this case, there is no variation of the word.

It is important to note that, if the tag IN is found in this part, everything before it is ignored as, beyond the author's name, it is not relevant information. Also, punctuation is ignored as it is used, at most, to divide sentences. Only quotation marks and colons are accepted.

If any string is still not found by the program, it is added with the keyword "other" to the citation array. This is important because we want to make sure that nothing is ignored because of computer mistakes and can always be used afterwards.

In the end, we end up with the citation metadata, which is an array with the authors' name, publication's title, pages, notes, volumes, year, editors, and editions. All of this information is then added to the authors' database (Table 3.4).

5.4 Summary

In this chapter, we successfully extracted citation information by employing grammar rules to segment tokens and subsequently parsed this information for storage in the database. Through this process, we gained valuable insights into the utility of rules, particularly with the assistance of regular expressions (regex). Furthermore, by utilizing chunking techniques, we were able to conveniently store specific components of the citation, further enhancing the accuracy and efficiency of the extraction process.

This chapter successfully answers question four introduced in chapter 1, "*How to identify the metadata elements characterizing a citation in order to build a list with all references associated with a specific work?*".

Chapter 6

Results

This chapter presents and discusses the results gathered for the different citation identification approaches that were followed in this work. We also explore some of BERT’s capabilities to be used in the project and how the information extracted from the citations identified can be displayed by a future developer.

The approach followed in the implementation of tools for the different stages of the project was accompanied by unit tests. Unit tests contribute to better validity and reliability of the research findings and thus provide reinforced confidence in the accuracy and functionality of the developed approaches. We used a pool of superior court documents that after preprocessing and data cleaning were the input of a learning algorithm to identify citations. Naturally, the initial pool of curated documents was partly used for learning and part for validation, thus avoiding overfitting, based on which metrics were calculated and reported as results.

At the end of this chapter, we present a possible solution to visualize the citation metadata extracted, as described in the last chapter. This concludes the project’s pipeline, as shown in figure 6.1.



Figure 6.1: Project’s Pipeline: Information Presenting Stage

6.1 Citation Identification Results Analysis

In this section, we compare the different approaches made to the citation identification algorithm, without the use of transformative architecture.

6.1.1 Identification based on patterns and researcher's names

As stated previously in chapter 4, an output file was created, not only for BERT training but for values comparison. The team made an effort to validate manually if any false positives or negatives were appearing in the code and unfortunately, some were. In the 7382 cases throughout all 20 documents lines evaluated, this is our balance:

- True Negatives: 7256;
- True Positives: 64;
- False Negatives: 57;
- False Positives: 5.

At first glance, these numbers do not look great, so we calculated the validation parameters for these values:

- Accuracy: 99,16%;
- Precision: 92,75%;
- Recall: 52,89%;
- Specificity: 99,93%.

We are aware that the high discrepancy between these numbers is due to the large number of True Negatives values that we have compared with the False Negatives. Where it really fails is on the recall because the formula compares the True Positives and the False Negatives. This indicates that roughly 47% of all citations are not correctly identified and that is very concerning! So there was a need to look through all False Negatives to understand what was going on.

After inspecting file by file in the 15 files that had an FP or FN, we gathered the main problems:

- In 14 out of the 16 files, the main problem was that the name was not on the researchers' list. A complete list could show that 9 out of these 15 problematic files could be fully correct;
- In 3 out of the 16 files, the main problem was because a shorter version, or an alias, of the researcher's name, was used. This cannot be accepted because, for example, the referenced name was "Vaz Serra" but in the researchers list we have 2 different researchers with these two names inside of their full names;
- In 2 out of the 16 files, the tokenization had problems and some sentences had been merged together without a separation space being added. This is a problem with the origin in the HTML conversion of DCSI website pages;

- In 2 out of the 16 files, the identified citations did not have a name correlated with the publication name, so they can be labelled as wrong citations.
- In 1 file, 3 names were mistakenly identified.

By removing the False Negatives that resulted in known researchers not having a match, this value was reduced by 42 instances, from 57 to 15. The problems with having smaller names or aliases are that firstly the longer names can contain shorter ones inside of them, which can mislead by identifying the wrong author, but the extraction stage should correctly take care of this. The second problem is that sometimes smaller names are used in the middle of a normal sentence to address them as a redactor and not as a publication's author. This is mainly because of the types of patterns that were gathered.

After this manual verification, we added the missing authors' names to the researcher's list and got the following values (More details on Table A.6):

- True Negatives: 7247;
- True Positives: 106;
- False Negatives: 15;
- False Positives: 14;
- Accuracy: 99,61%;
- Precision: 88,33%;
- Recall: 87,60%;
- Specificity: 99,81%.

The DGSI website had HTML issues that made the web scrapper create weird glitches when parsing the data. The majority of these were corrected in the data preparation data but problems such as the missing space after the period punctuation, random parentheses in the middle of the phrase that would disrupt the tokenization process, or even phrases such as the following that would sometimes appear:

```
n em co m a a utonomia ou in dependência carac terística d os  
reg ulamentos d os plan os muni cipais
```

After a simple cleaning of some phrases that had abnormalities like this one, and after removing the patterns that were causing the second problem mentioned, we got the following values (More details on Table A.7):

- True Negatives: 7255;

- True Positives: 103;
- False Negatives: 19;
- False Positives: 5;
- Accuracy: 99,67%;
- Precision: 95,37%;
- Recall: 84,43%;
- Specificity: 99,93%.

These are great results, but we have to be thoughtful about them. The decline in False Positives is due to the fact that fewer patterns are used, making it so that fewer patterns are misused. But this way more False Negatives are created, because a great portion of the sentences have uncommon citations that rely on those patterns to be identified, for example:

No campo doutrinário, como dá conta Carla Gonçalves,
"há quem defenda que a distinção entre a obrigação
de meio e a de resultado vem perdendo importância."

The higher number of False Negatives, compared to a similar number of True Positives, leads to a decrease in recall. Conversely, the decrease in False Positives to a similar number of True Positives results in an increase in precision.

6.1.2 Identify by grammar rules

Identifying citations by adopting grammar rules has its pros and cons. On one hand, they are easier to implement as we only have to come up with a rule for the citation but then, on the other hand, it can be harder to come up with a rule that accommodates all different formats a citation may have and thus may identify many False Positives. For this reason, we created a rule based on the grammar built on chapter 5, and toned it down to be more precise. The results can vary based on the rule and on the authors' list.

```
NAME: {<NPROP2>((<NPROP>|<PREP>)?<NPROP2>)+}
PAG_DOT: {<PAG><PONT>?}
PAGS: {<PAG_DOT><NUM>((<PUNC>|<KC>|<ART>)<NUM>)*(<KC><SGS>)?}
REF: {<NAME><.*>+?<PAGS>}
```

For the authors' list, we used the same approach as presented in the chapter 5, which is a list of unique names throughout authors. The results obtained were the following:

- True Negatives: 7251;
- True Positives: 95;
- False Negatives: 26;
- False Positives: 10;
- Accuracy: 99,51%;
- Precision: 90,48%;
- Recall: 78,51%;
- Specificity: 99,86%.

Even though this method is worst from a recall perspective compared to the last section's procedure, it is more flexible as it does not need the researcher's list, it only requires unique singular names from the database. Figure 6.2 compares these validators' metrics. More details are provided in tables A.5, A.6, and A.9.

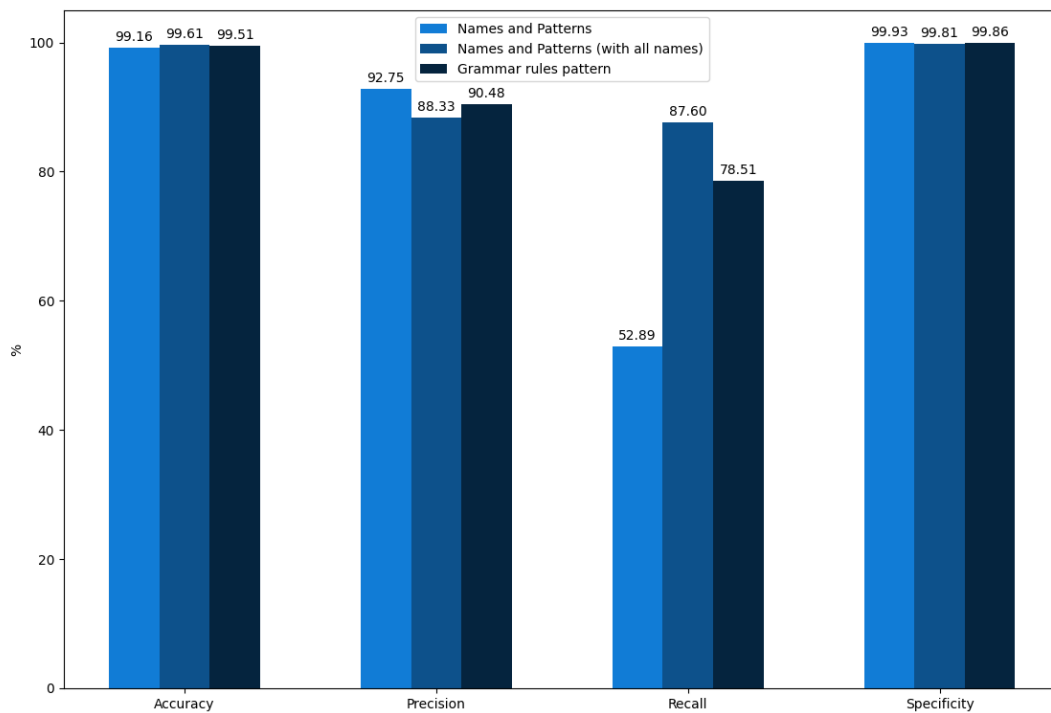


Figure 6.2: Comparison of validators metrics between different approaches on citation identification

6.1.3 BERT

BERT, as a machine learning tool, is very dependent on the quality of the dataset. Legal documents have commonly long phrases. In figure 6.3 we can see the distribution of these phrases, after removing outliers.

In our analysis, sentences below a length of 50 characters were excluded from consideration. This decision was made based on the presence of citations in sentences, as we assumed that shorter sentences are less likely to contain citations. The lower quartile in the box plot calculation indicated that sentences with a length of 56 characters or longer were included in the dataset (as can be seen in Figure B.2). Many outliers by greatness appeared after 1000 characters, so we blocked any phrase with a length above 1000. With these values, the median of all phrases, as seen in figure 6.3, is 199, being the lower quartile 125 and the upper one 309.

Since 512 tokens, which is the maximum available for BERT, is roughly equivalent to 2500 characters [53], this is for English vocabulary, we decided to go with a max length of 64 tokens on BERT. If you follow the English conversion, 200 characters would be equivalent to almost 41 characters. We wanted to go by excess so we do not come short.

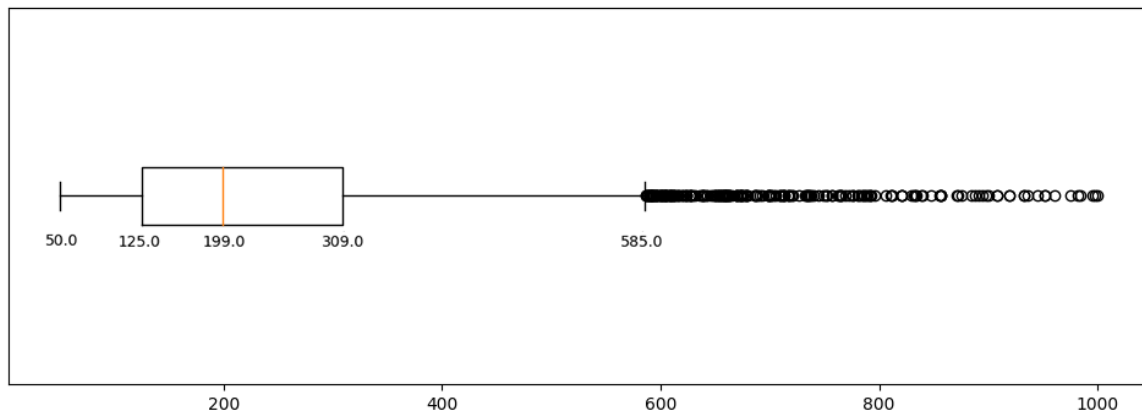


Figure 6.3: Boxplot on sentence length distribution for all 7434 lines gathered

With this being said, to make sure that BERT was well explored as a possible solution, an effort was made to positively enhance the quality of the dataset. In the end, the best values for the variables were for epoch with value 2 (we will see it deeper later in this section), learning rate at $5e-5$, and batch size at 16. The Graph 6.4, and 6.5 express this efforts.

The first iteration had all the lines that were found, disregarding the last calculations and independent of length or quality. The bad results on this can be translated into not being a really good fine-tuning, so it just guessed if the phrase was a sentence with a citation or not, based on a negative percentage on the dataset.

The second iteration had deeper hand-picked sentences gathered. It only accepted sentences

with a minimum length of 50 characters and a maximum of 1000. Also, each negative sentence would have a 60% chance of being removed. The final dataset had 1338 negative sentences and 58 positives.

The third and final iteration had a tighter selective length, with a minimum of 100 and a maximum of 700. The negative sentences had a 30% chance of survival during the purge. The final dataset had 1718 negatives and 58 positives. These size values were not random, as they were selected based mainly on the largest and shortest positive sentence.

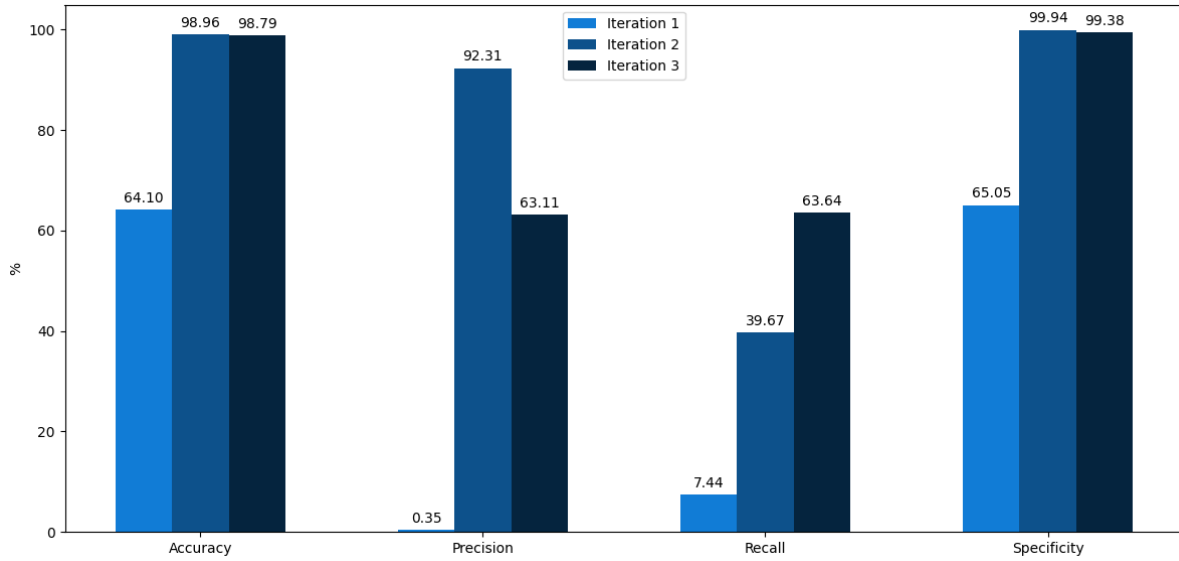


Figure 6.4: Comparison of validator's metrics between different iterations of BERT's dataset

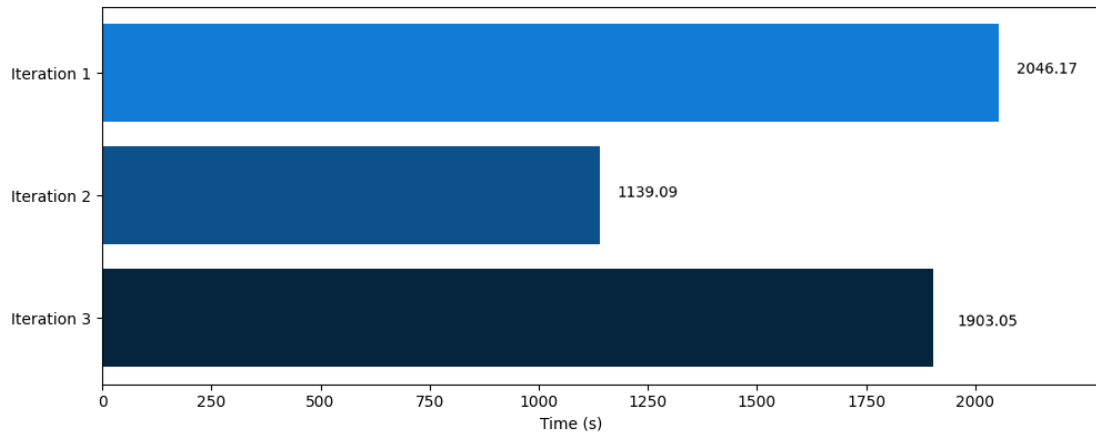


Figure 6.5: Comparison of computational times between different iterations of BERT's dataset

Different epochs were made to the third iteration. Using epoch at level 3 made precision higher but recall lower. Using epoch at 4, made everything worse, so we are better at fine-tuning the model with 2 epochs and all previous variables. Figure 6.6 presents the comparison of the validator's metrics throughout different epochs in the same dataset.

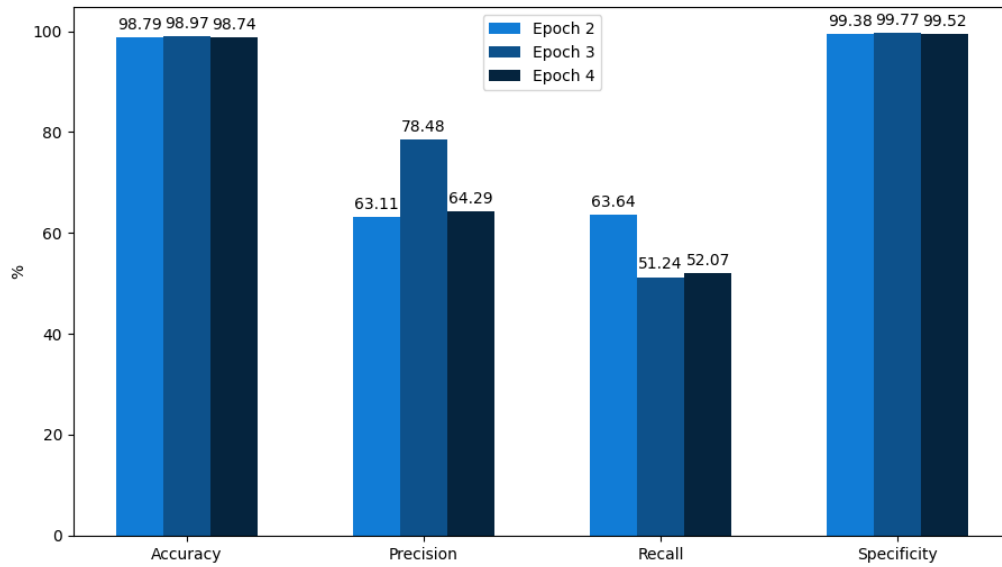


Figure 6.6: Comparison of validator's metrics between different epochs' values on the same iteration of BERT's dataset

Upgrading the number of negatives and maintaining the number of positives, with a controlled selection, provided a lower precision but a higher, above 50% (which could be considered true randomness) recall. Regrettably, the second question posed in Chapter 1, "Can this identification process benefit from machine learning capabilities?" is not conclusively answered in this section.

Our results do not demonstrate an improvement compared to a traditional algorithm that relies on patterns and authors' names. However, based on them, we can expect that a larger and higher-quality dataset may yield considerable improvements in citation identification, potentially eliminating the need for authors' names or specific patterns to identify it. This suggests that leveraging machine learning techniques and expanding the dataset could offer promising avenues for future research in enhancing citation identification methods.

6.1.4 Conclusions

By comparing results, times, and complexity, we can conclude that, if a future iteration of this work wants to be limited to a pre-constructed list, to maintain reliability on the system, the best approach would be the one with patterns and authors' names. If not, the researcher should make a choice between the grammar ruling and BERT. On one hand, it is easier to implement and control, and the overall times are quicker. On the other hand, BERT might increase its positive identification even further by having a greatly populated dataset. Figure 6.7 compares the identification times between the three iterations.

It is important to note that the times that were displayed for BERT iterations do not regard

the training time. As that is a separate action, we wanted it to be as reliable as possible to be compared to other approaches.

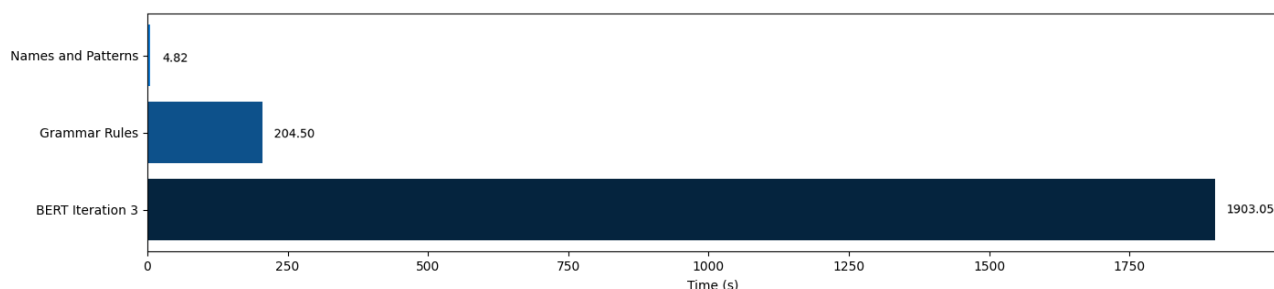


Figure 6.7: Comparison of times to identify citations in 20 documents between different approaches to the citation identification problem

6.2 Presenting extracted data

From the information that we extracted in chapter 5, we got the following metadata, if available: Authors' names, Publication Title, Pages, Volumes, Notes, Year, Editor, Edition, and Others. To present a possible output for the project, we generated stylized references having the following rules:

- The authors' name should start with the last one, then a comma;
- Then the rest of the name starting with the beginning;
- Then, if available, the year inside parenthesis;
- A period is made to open quotation marks to the publication name and then close those quotation marks;
- A comma is introduced to make pages available. The word "págs." or "pág." (if singular) is added, followed by each number of pages, separated by a comma. If the "+" sign exists in the pages array, the word "sgs." is added, expressing "and following";
- The same logic is added to volumes and notes, but the preceding words used are "vols." and "notes", respectively;
- It then adds a period followed by the editor, edition, and every other part that was not identified, all of this separated by a period.

The following example might help understand this better:

Information fetch from the database: `[[author: 'Maria Raquel Guimarães', publication: 'As Transferências Eletrônicas De Fundos E Os Cartões De Débito', pags: '[11, 12, '+]', year: 1999, other: 'Almedina']]`

Output: *Guimarães, Maria Raquel (1999). "As Transferências Eletrônicas De Fundos E Os Cartões De Débito", págs. 11, 12 e sgs.. Almedina.*

In this section and with the information stated in chapter 3, we were also able to answer the third question introduced in chapter 1, "*How to credit researchers and law practitioners with citations to their work found in superior court decision documents?*".

6.3 Summary

This chapter described the results of the work carried out in this dissertation. The main focus of this dissertation was achieved as we can see by the results of the citation identification by pattern and authors' names. The combined use of these two methods yielded validation metrics that surpassed other approaches. While the implementation of BERT did not yield satisfactory results, we remain optimistic about its potential for improvement, as discussed in the subsequent chapter, on future work.

The effective presentation of extracted information in a typical reference style demonstrates its utility and potential applications for future developers. This standardized format allows for versatile use in various contexts and enhances the accessibility and usability of the extracted data.

Chapter 7

Conclusion and Future work

In this dissertation, we presented a strategy to deal with citation identification in superior court decision documents. It includes three parts: (1) Data preparation, where decision court documents are preprocessed into a format manageable by our algorithms; (2) citation identification, where we presented different algorithm solutions to identify citations to legal documents and evaluate them on a validated dataset to discover their differences and uniqueness; and (3) citation extraction where used grammar syntax on Portuguese sentences to wrap chunks of words and extract different kinds of information to the database.

The approaches used to identify citations were: identification based on patterns and researcher's names, identification based on grammar rules, and using BERT. Our results show that a traditional algorithm based on patterns and author's names achieved the best identification results, however, if we could resort to a larger curated dataset on which a learning algorithm could learn, the conclusion could be slightly different.

7.1 Future Work

Our work was an initial approach to the problem of citation identification in legal documents and thus further work will be necessary to extend and improve its functionality and implementation. Firstly, citation identification from techniques that do not use transformer architecture can benefit from deep refactoring. Extending the optional punctuations, sentence tokenization, and reference patterns can have a great performance increase.

Altering the authors' names listing for citation identification can also be a great addition. Another algorithm making use of Redis as a message broker and a cache database could be adopted to keep track of the most commonly searched, used, or stored names in the database. In this case, even if the time complexity would still be at $O(n)$ for the author name search, we can still, in the long run, have better results. We could also just store the first name of the authors' and then, if found, we could look for all of the names under that first name. It needs to be pointed out that this can lead to hard code decisions because, as we said before, names can be displayed

in different ways and first names do not necessarily mean that they are the first to be displayed.

One critical approach is to relate the names to the sentence but make sure that they are next to each other. For one phrase that has two names to two different citations, our implementation shows that if it exists a third individual can be wrongly identified, based only on those two names.

Considering that BERT is an Artificial Intelligence model, it will always benefit from large datasets with great quality. Having this in mind, a greater and wider dataset with good, different, positive citations on phrases can help our model to become better. It is crucial to also create a model for BERT that is fine-tuned for legal literature. This can eventually help the tokenization process and help the model learn legal knowledge and terminology, leading to more accurate predictions on these tasks. Our model had 7434 sentences but only 121 were positives, which is only almost 2% of the dataset. Future research should increase this proportion to aim for better results, as for our best solution, we tried to make it 10% by cutting down on negatives.

Citation extraction can be fine-tuned to include other parts of citations that were not seen. Rules can be enhanced and improved to the developer's necessity or if needed. Also, a future researcher can try to include a transformative AI with named entity recognition capabilities to extract these types of information.

Lastly, this work can also be used as a blueprint for other languages. Citation and extraction would be very similar as it uses grammar rules to identify words. Advice for future work to be done on other languages is to look deeper into NLTK's "Extracting Information from Text" chapter as chunk's use cases can go much deeper as we go on this project, for example, chunk evaluation, chunking, and regex tag pattern for rules with sub-token for each word.

It is also important to note that some results of this dissertation took a hit on their validation because many parts of the extracted information from the DGSI's website were not completely correct (as seen in previous chapters). If they are correctly fetched and then parsed, tokenization will be clear and the results would be much better.

In summary, improving the identification of unique citations has a twofold benefit. Firstly, it enhances the quality of datasets, enabling the identification of more specific patterns and information. This, in turn, streamlines the process of citation identification, reducing computational complexity and making it a more straightforward task. By focusing on identifying unique citations, researchers can optimize the efficiency and accuracy of their work, ultimately contributing to the advancement of the field.

Appendix A

Auxiliary Tables

Table A.1: HTML Requests Header for Web Scrapping Algorithm

Key	Value
User-Agent	Mozilla/5.0 (Windows NT 10.0; Win64; x64) AppleWebKit/537.36 (KHTML, like Gecko) Chrome/106.0.0.0 Safari/537.36
Encoding	ISO-8859-1

Table A.2: Example of patterns in european portuguese legal literature for citations

in	by	pag.	pág.
pags.	págs.	p.	pp.
pps.	cfr.	citando	afirma que
a expressão é de	como [...] sub- linha	como dá conta	como ensina
como nos dá conta	como sublinha	como sugerido por	como sustenta
conforme alerta	conforme dis- corre	conforme explicam	conforme explica
do mesmo modo,	em sentido próximo [...] se pronuncia	escreve	explica o autor
na doutrina	na síntese de	nas palavras de	neste sentido
neste sentido, afirma	no mesmo sen- tido aponta	no mesmo sentido aponta também o prof.	No mesmo sen- tido, refere [...] in artigo pub- licado
para	segundo	Refere, a este propósito,	seguindo [...] a obra de
sobre [...] vide	sobre este ponto, [...] vide ainda	neste sentido, vide	em sentido oposto, vide
sobre o tema pode ver-se ainda	vide, por todos	como acentua a	como acentua o
por Acórdão do	por Acórdão da	mencionado acórdão do	mencionado acórdão da

Table A.3: Mac-Morpho's lexical tags and its grammar class

Grammar Class	Tag
ADJETIVE	ADJ
ADVERBS	ADV
ADVERBIAL CONNECTIVE SUBORDINATIVE	ADV-KS
RELATIVE ADVERBIAL SUBORDINATIVE	ADV-KS-REL
ARTICLE (def. or indef.)	ART
COORDINATING CONJUNCTION	KC
SUBORDINATING CONJUNCTION	KS
INTERJECTION	IN
NAME	N
PROPER NOUN	NPROP
NUMERAL	NUM
PARTICLE	PCP
DENOTATIVE WORD	PDEN
PREPOSITION	PREP
ADJECTIVE PRONOUN	PROADJ
CONNECTIVE PRONOUN SUBORDINATIVE	PRO-KS
PERSONAL PRONOUN	PROPESS
RELATIVE CONNECTIVE PRONOUN SUBORDINAT- IVE CONNECTIVE	PRO-KS-REL
PRONOUN NOUN	PROSUB
VERB	V
AUXILIARY VERB	VAUX
CURRENCY SYMBOL	CUR
COMPLEMENTARY LABELS (Foreignisms; Appositives; Data; Telephone Numbers; Dates; Times; and Disjunction)	EST AP DAD TEL DAT HOR []
CONTRACTIONS and EN- CONCLISES	+
MESOCLES	!

Table A.4: Example of paires "Word-Tag" used in the custom Mac-Morpho list

Key	Value
pag.	PAG
pags.	PAG
pág.	PAG
págs.	PAG
vol	VOL
/	PUNC
,	PUNC
(	PUNC
)	SEP
;	SEP
.	PONT
Fábio	NPROP2
cit.	N
in	IN
sgs	SGS
ss	SGS

Table A.5: Citation identification from authors' names and patterns (Raw)

ID	TN	TP	FN	FP
1	91	3	7	0
2	254	0	0	0
3	272	3	0	0
4	443	0	1	0
5	250	3	5	0
6	195	7	2	0
7	348	7	1	0
8	509	3	1	0
9	149	1	1	0
10	546	1	0	0
11	360	4	0	0
12	299	3	3	0
13	789	4	0	0
14	272	3	6	0
15	363	5	0	0
16	436	2	8	0
17	206	2	6	1
18	725	3	5	0
19	255	3	2	4
20	282	1	1	0
21	212	6	8	0
Total	7256	64	57	5

Table A.6: Citation identification from authors' names and patterns (All names)

ID	TN	TP	FN	FP
1	91	9	1	0
2	254	0	0	0
3	272	3	0	0
4	443	0	1	0
5	250	8	0	0
6	195	8	1	0
7	348	8	0	0
8	509	3	1	0
9	148	2	0	1
10	546	1	0	0
11	360	4	0	0
12	299	6	0	0
13	789	4	0	0
14	272	8	1	0
15	363	5	0	0
16	434	9	1	2
17	206	2	6	1
18	722	8	0	3
19	254	4	1	5
20	282	2	0	0
21	210	12	2	2
Total	7247	106	15	14

Table A.7: Citation identification from authors' names and patterns (All names and less Patterns)

ID	TN	TP	FN	FP
1	91	9	1	0
2	254	0	0	0
3	272	3	0	0
4	443	0	1	0
5	250	8	0	0
6	195	8	1	0
7	348	7	1	0
8	509	4	0	0
9	149	2	0	0
10	546	1	0	0
11	360	4	0	0
12	299	6	0	0
13	789	4	0	0
14	272	8	1	0
15	363	4	1	0
16	434	9	1	2
17	206	2	6	1
18	724	7	1	1
19	258	3	2	1
20	282	1	1	0
21	211	13	2	0
Total	7255	103	19	5

Table A.8: Citation identification from authors' names (All names), and Grammar Rules

ID	TN	TP	FN	FP
1	91	9	1	0
2	254	0	0	0
3	272	3	0	0
4	443	0	1	0
5	250	7	1	0
6	195	8	1	0
7	347	6	2	1
8	509	3	1	0
9	149	2	0	0
10	546	1	0	0
11	360	4	0	0
12	299	6	0	0
13	789	4	0	0
14	272	6	3	0
15	363	4	1	0
16	435	9	1	1
17	207	0	8	0
18	724	7	1	1
19	259	3	2	0
20	282	1	1	0
21	210	10	5	1
Total	7256	93	29	4

Table A.9: Citation identification from Grammar Rule's Patterns

ID	TN	TP	FN	FP
1	91	9	1	0
2	254	0	0	0
3	272	3	0	0
4	443	0	1	0
5	250	7	1	0
6	194	9	0	1
7	347	6	2	1
8	509	3	1	0
9	149	2	0	0
10	546	1	0	0
11	360	4	0	0
12	299	6	0	0
13	789	4	0	0
14	271	7	2	1
15	362	4	1	1
16	435	9	1	1
17	207	0	8	0
18	724	7	1	1
19	259	4	1	0
20	280	1	1	2
21	210	9	5	2
Total	7251	95	26	10

Table A.10: Citation identification from BERT, iteration 1

ID	TN	TP	FN	FP
1	73	2	8	18
2	158	0	0	96
3	185	0	3	87
4	305	0	1	138
5	166	2	6	84
6	152	0	9	43
7	201	1	7	147
8	368	1	3	141
9	109	1	1	40
10	414	0	1	132
11	165	0	4	195
12	220	0	6	79
13	607	0	4	182
14	170	0	9	102
15	144	0	5	219
16	253	2	8	183
17	113	0	8	94
18	403	0	8	322
19	164	0	5	95
20	191	0	2	91
21	162	0	14	50
Total	4723	9	112	2538

Table A.11: Citation identification from BERT, iteration 2

ID	TN	TP	FN	FP
1	91	2	8	0
2	254	0	0	0
3	272	1	2	0
4	443	0	1	0
5	250	3	5	0
6	195	7	2	0
7	347	5	3	1
8	509	4	0	0
9	149	1	1	0
10	546	1	0	0
11	360	3	1	0
12	299	3	3	0
13	789	3	1	0
14	272	2	7	0
15	363	5	0	0
16	436	2	8	0
17	207	0	8	0
18	725	1	7	0
19	256	1	4	3
20	282	1	1	0
21	212	3	11	0
Total	7257	48	73	4

Table A.12: Citation identification from BERT, iteration 3

ID	TN	TP	FN	FP
1	89	1	9	2
2	254	0	0	0
3	271	2	1	1
4	443	1	0	0
5	249	8	0	1
6	194	5	4	1
7	346	7	1	2
8	505	3	1	4
9	148	2	0	1
10	543	1	0	3
11	359	4	0	1
12	298	3	3	1
13	785	3	1	4
14	269	7	2	3
15	358	5	0	5
16	435	8	2	1
17	204	0	8	3
18	722	5	3	3
19	253	4	1	6
20	281	1	1	1
21	210	7	7	2
Total	7216	77	44	45

Appendix B

Auxiliary Graphs

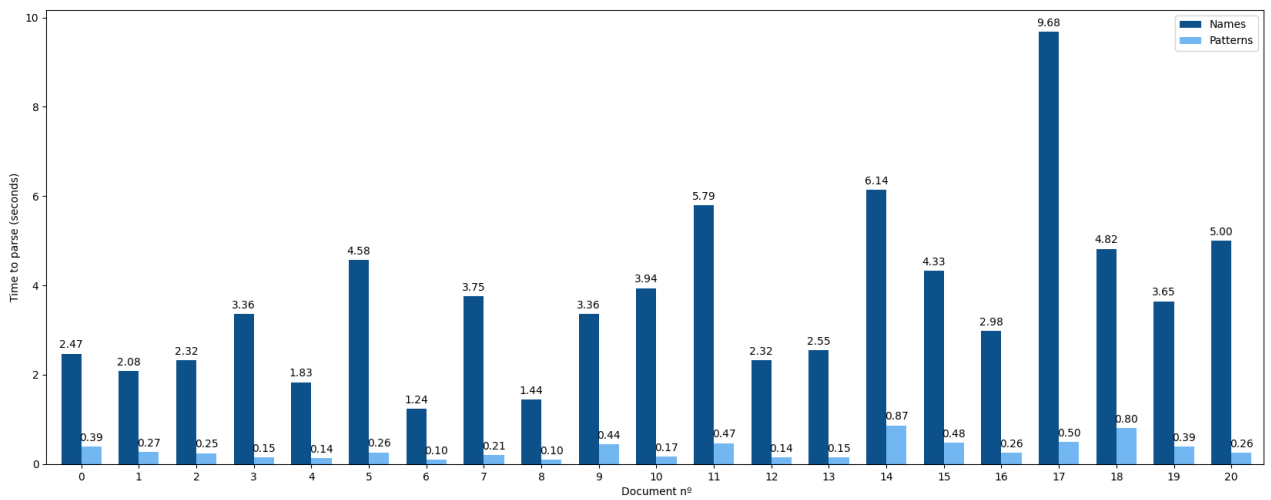


Figure B.1: Graph plot relating identification via patterns first and names first.

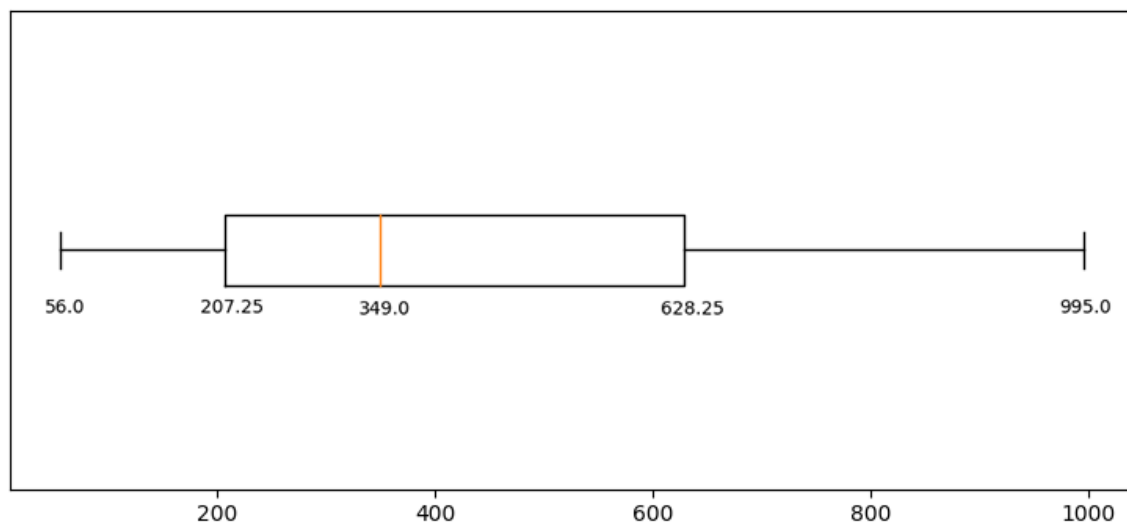


Figure B.2: Boxplot on sentence length distribution for all 121 lines with citation gathered

Appendix C

Auxiliary Code Samples

Listing C.1: BERT Training

```
1 #In the optimizer, lr, meaning learning rate, can and should be changed on necessity
  optimizer = torch.optim.AdamW(model.parameters(), lr=5e-5, eps=1e-08)
3 # Initialize device to use GPU if possible
  device = torch.device('cuda' if torch.cuda.is_available() else 'cpu')
5 epochs = 2

7 for _ in range(epochs, desc = 'Epoch'):

9     # Set model to training mode. This model in our case was the base BERTimbau
    model.train()

11
    # Initialize tracking variables
13     tr_loss = 0
    nb_tr_examples, nb_tr_steps = 0, 0

15
    for step, batch in enumerate(train_dataloader):
17         batch = tuple(t.to(device) for t in batch)
        b_input_ids, b_input_mask, b_labels = batch
19         optimizer.zero_grad()
        # Forward pass
21         train_output = model(b_input_ids,
                               token_type_ids = None,
23                               attention_mask = b_input_mask,
                               labels = b_labels)

25         # Backward pass
        train_output.loss.backward()
27         optimizer.step()
        # Update tracking variables
29         tr_loss += train_output.loss.item()
        nb_tr_examples += b_input_ids.size(0)
31         nb_tr_steps += 1
```

Listing C.2: BERT Validation

```
# Set model to evaluation mode
2 model.eval()

4 # Tracking variables
  val_accuracy = []
6 val_precision = []
  val_recall = []
8 val_specificity = []
```

```

10 for batch in validation_dataloader:
11     batch = tuple(t.to(device) for t in batch)
12     b_input_ids, b_input_mask, b_labels = batch
13     with torch.no_grad():
14         # Forward pass
15         eval_output = model(b_input_ids,
16                             token_type_ids = None,
17                             attention_mask = b_input_mask)
18     logits = eval_output.logits.detach().cpu().numpy()
19     label_ids = b_labels.to('cpu').numpy()
20     # Calculate validation metrics
21     b_accuracy, b_precision, b_recall, b_specificity = b_metrics(logits, label_ids)
22     val_accuracy.append(b_accuracy)
23     # Update precision only when (tp + fp) != 0; ignore nan
24     if b_precision != 'nan': val_precision.append(b_precision)
25     # Update recall only when (tp + fn) != 0; ignore nan
26     if b_recall != 'nan': val_recall.append(b_recall)
27     # Update specificity only when (tn + fp) != 0; ignore nan
28     if b_specificity != 'nan': val_specificity.append(b_specificity)
29 # Then model can be saved for future use

```

Listing C.3: BERT Usage

```

# Auxiliary method. It returns BatchEncoding with a list of token ids,
# list of token type ids, an attention mask list
def preprocessing(input_text, tokenizer):
4     return tokenizer.encode_plus(
5         input_text,
6         add_special_tokens=True,
7         max_length=max_length,
8         padding='max_length',
9         return_attention_mask=True,
10        return_tensors='pt',
11        truncation=True
12    )

14 # Model should be loaded before
15 # We need Token IDs and Attention Mask for inference on the new sentence
16 test_ids = []
17 test_attention_mask = []
18 # Apply the tokenizer
19 encoding = preprocessing(phrase, tokenizer)
20
21 # Extract IDs and Attention Mask
22 test_ids.append(encoding['input_ids'])
23 test_attention_mask.append(encoding['attention_mask'])
24 test_ids = torch.cat(test_ids, dim=0)
25 test_attention_mask = torch.cat(test_attention_mask, dim=0)
26
27 # Forward pass, calculate logit predictions
28 with torch.no_grad():
29     output = model(test_ids.to(device), token_type_ids=None,
30                   attention_mask=test_attention_mask.to(device))
31
32 # It will return "Citation" if it identifies a citation and "Normal" if not
33 prediction = "Citation" \
34     if np.argmax(output.logits.cpu().numpy()).flatten().item() == 1 \
35     else "Normal"

```

Appendix D

Auxiliary Images

Acórdão STJ		Acórdão do Supremo Tribunal de Justiça	
Processo: N.º Conventional: Relatório: Descrições:	III/II.LITR.VR.PLSI 1.ª SECÇÃO Exemplo de Nome REGISTO AUTOMÓVEL DOCUMENTO PARTICULAR PODERES DO SUPREMO TRIBUNAL DE JUSTIÇA PRESUNÇÃO DERIVADA DO REGISTO ONIS DA PROVA NULIDADE DO REGISTO POSSE PRESUNÇÃO DE POSSE		
Data do Acórdão: Votação: Texto Integral: Privacidade: Muito Interessante: Decisão: Área Temática:	01-01-2019 UNANIMIDADE S I REVISTA NEGADAA REVISTA DIREITO CIVIL - DIREITO DAS COISAS / POSSE / DISPOSIÇÕES GERAIS / NOÇÃO / EXERCÍCIO DA POSSE POR INTERMEDIÁRIO.		
Doutrina:	Carvalho Fernandes, Lições de Direitos Reais, 6.ª edição, Quil Juris, Lisboa, 2009, p. 144; Hériz, A função do registo como meio de protecção do tráfico jurídico, Regista, n.º 70, 1986, p. 276; Isabel Pereira Mendes, Código do Registo Predial Anotado e Comentado, 17.ª edição, 2009, p. 178; Manuel de Andrade, Teoria Geral da Relação Jurídica, Vol. I, Facto Jurídico, em especial Negócio Jurídico, 8.ª, reimpressão, Coimbra, 1998, p. 19-20; Maria Clara Sottomayor, Invalidez e registo, A protecção do terceiro adquirente de boa fé, 2010, Almedina, Coimbra, p. 720 e 863; Orlando de Carvalho, Introdução à posse", RLJ n.º 3780 e ss, in Direitos das Coisas, Coordenação de Liberal Fernandes/Raquel Guimarães/Regina Redinha, Coimbra Editora, 2015, p. 268; Var Serra, Anotação ao Acórdão do STJ de 12 de julho de 1963, RLJ, Ano 97, 1964-65, n.º 3265, p. 57.		
Legislação Nacional:	CÓDIGO CIVIL (CC) - ARTIGOS 1251.º, N.º 1 E 1252.º, N.º 2; CÓDIGO DE REGISTRO PREDIAL (CRPRED) - ARTIGOS 7.º, 16.º, N.º 1, ALÍNEA B) E 17.º, N.ºS 1 E 2.		
Jurisprudência Nacional:	ACÓRDÃO DO SUPREMO TRIBUNAL DE JUSTIÇA: ACÓRDÃO UNIFORMIZADOR DE JURISPRUDÊNCIA, DE 14-05-1996; ACÓRDÃO UNIFORMIZADOR DE JURISPRUDÊNCIA, DE 18-05-1999; DE 17-12-2015, PROCESSO N.º 940/16.9TYPRP.PLSI; DE 07-02-2017, PROCESSO N.º 9071/15.6TYVNEGLSI; DE 21-06-2018, PROCESSO N.º 4500/15.9TYCBRC.LS2; DE 07-02-2019, PROCESSO N.º 2916/15.9TYBVD.LLS2.		
Sumário :	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec quis pulvinar sapien. Nulla ut dui lectus. Praesent convallis urna sit amet velit imperdiet porttitor. Sed nec purus nec erat scelerisque rhoncus vel eget tortor. Pellentesque vel vulputate magna, eget pretium ex. Proin porttitor ipsum eleifend pharetra lacinia. Nullam interdum metus sed justo vestibulum, a malesuada sem tempus. Proin sed elementum sapien. Nullam ac condimentum dui, pellentesque malesuada ex. In in magna mollis mi aliquet tincidunt. Phasellus vel cursus ex. Proin convallis sapien nisl. Fusce semper sollicitudin convallis. Curabitur dapibus interdum massa id vehicula. In hac habitasse platea dictumst. Vestibulum sagittis vitae dolor sed congue. Ut ut condimentum sem. Pellentesque ac sollicitudin neque. Etiam pellentesque congue lobortis. Pellentesque mattis nulla ac mauris efficitur semper. Proin maximus, mauris vel fermentum imperdiet, elit lacus posuere nisl. ut porttitor mauris nulla ut nulla. Suspendisse ac viverra nulla. Suspendisse vel hendrerit nulla, ac scelerisque lorem. Integer vehicula rhoncus urna sit amet varius. Maecenas tincidunt purus ex, ac tempor nisl pellentesque vel. Integer at dapibus nisi, in suscipit sapien. Nulla in dolor purus. Phasellus at porttitor turpis. Donec faucibus nunc finibus, rhoncus leo eu, congue quam. Nam imperdiet congue tempor. Nam sem eros, auctor eu magna et, scelerisque sollicitudin leo. Donec non metus ut nisl lobortis sagittis quis in magna. Donec enimod leo et mauris molestie, at fringilla nulla interdum. Aenean vestibulum nunc et dapibus egestas. In non libero mollis, dignissim lorem vel, condimentum urna. Nullam enim augue, consectetur rutrum scelerisque id, accumsan quis augue. Fusce scelerisque sit amet nunc id tempor.		
Decisão Texto Integral:	Lorem ipsum dolor sit amet, consectetur adipiscing elit. Donec quis pulvinar sapien. Nulla ut dui lectus. Praesent convallis urna sit amet velit imperdiet porttitor. Sed nec purus nec erat scelerisque rhoncus vel eget tortor. Pellentesque vel vulputate magna, eget pretium ex. Proin porttitor ipsum eleifend pharetra lacinia. Nullam interdum metus sed justo vestibulum, a malesuada sem tempus. Proin sed elementum sapien. Nullam ac condimentum dui, pellentesque malesuada ex. In in magna mollis mi aliquet tincidunt. Phasellus vel cursus ex. Proin convallis sapien nisl. Fusce semper sollicitudin convallis. Curabitur dapibus interdum massa id vehicula. In hac habitasse platea dictumst. Vestibulum sagittis vitae dolor sed congue. Ut ut condimentum sem. Pellentesque ac sollicitudin neque. Etiam pellentesque congue lobortis. Pellentesque mattis nulla ac mauris efficitur		

Figure D.1: DGSI's URL Structure

References

- [1] UNSW - Sidney, “Why is referencing important?.” www.student.unsw.edu.au/why-referencing-important Last accessed Mar. 20 2023.
- [2] “IGFEJ - Bases jurídico-documentais.” www.dgsi.pt, Last accessed Mar. 20 2023.
- [3] The Institute of Electrical and Electronics Engineers, Inc., “IEEE standard computer dictionary: A compilation of IEEE standard computer glossaries,” 1990.
- [4] P. C. (NCR), J. C. (SPSS), R. K. (NCR), T. K. (SPSS), T. R. (DaimlerChrysler), C. S. (SPSS), and R. W. (DaimlerChrysler), “Cross industry standard process for data mining,” 2000.
- [5] M. Juodyte, “Overview: Data mining pipeline.” Faculty of Informatics, Technical University of Munich, 2017.
- [6] J. Han, M. Kamber, and J. Pei, *Data mining concepts and techniques*. Morgan Kaufmann Series in Data Management Systems, 3rd edition ed., 2012.
- [7] R. K. Pearson, “The Problem of Disguised Missing Data,” *ACM SIGKDD Explorations Newsletter*, vol. 8, pp. 83–92, jun 2016.
- [8] K. Soo, “Deep learning bible,” 2023. Last edited on May 10 2023. wikidocs.net/book/8926. Last accessed Jun. 4 2023.
- [9] B. Dang, “The 8 best nosql databases for your business.” Datavid, datavid.com/blog/best-nosql-databases, Last accessed Jan. 11 2023.
- [10] S. Goled, “Self-supervised learning vs semi-supervised learning: How they differ,” in *Analytics India Mag*, May 2021. analyticsindiamag.com/self-supervised-learning-vs-semi-supervised-learning-how-they-differ. Last accessed Jun. 4 2023.
- [11] A. Radford, K. Narasimhan, T. Salimans, and S. Ilya, “Improving language understanding by generative pre-training.” https://s3-us-west-2.amazonaws.com/openai-assets/research-covers/language-unsupervised/language_understanding_paper.pdf, 2018.
- [12] O. Levy and Y. Goldberg, “Dependency-based word embeddings,” in *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 2: Short*

- Papers*), (Baltimore, Maryland), pp. 302–308, Association for Computational Linguistics, June 2014.
- [13] J. Pennington, R. Socher, and C. Manning, “GloVe: Global vectors for word representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, (Doha, Qatar), pp. 1532–1543, Association for Computational Linguistics, Oct. 2014.
- [14] J. Mannes, “Facebook’s fasttext library is now optimized for mobile.” TechCrunch, 2018. <techcrunch.com/2017/05/02/facebook-fasttext-library-is-now-optimized-for-mobile/> Last Accessed May 31 2023.
- [15] Radim Řehůřek, “Gensim: Topic modelling for humans.” <radimrehurek.com/gensim/>. Last accessed on June 13 2023.
- [16] Flair NLP, “Flair.” <github.com/flairNLP/flair>. Last accessed on June 12 2023.
- [17] “GROBID documentation.” <grobid.readthedocs.io/en/latest/>. Last accessed on June 12 2023.
- [18] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, “Attention Is All You Need,” in *31st Conference on Neural Information Processing Systems (NIPS 2017)*, Long Beach, CA, USA, Jun 2017.
- [19] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, “Exploring the limits of transfer learning with a unified text-to-text transformer,” *CoRR*, vol. abs/1910.10683, 2019.
- [20] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, 2018. <www.aclweb.org/anthology/N18-3012>. Last accessed on June 18 2023.
- [21] G. Ghati, “Comparison between bert, gpt-2 and elmo,” May 2020. medium.com/@gauravghati/comparison-between-bert-gpt-2-and-elmo-9ad140cd1cda Last Accessed June 27 2023.
- [22] S.-A. DeLucia, “Unleashing the power of bert: How the transformer model revolutionized nlp.” Website, March 2023. <arize.com/blog-course/unleashing-bert-transformer-model-nlp> Last Accessed May 20 2023.
- [23] S. Pichai, “An important next step on our ai journey.” Google AI Blog, Feb 2023. <blog.google> Last Accessed May 20 2023.
- [24] OpenAI, “Openai api documentation.” <platform.openai.com/docs> Last Accessed 24 Apr. 2023.

- [25] OpenAI, “Chatgpt.” <chat.openai.com> Last Accessed 24 Apr. 2023.
- [26] OpenAI, “DALL-E 2.” <labs.openai.com> Last Accessed 24 Apr. 2023.
- [27] K. Hu, “Chatgpt sets record for fastest-growing user base,” Feb. 2023. <www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01> Last accessed 24 Apr. 2023.
- [28] OpenAI, “GPT-4 is OpenAI’s most advanced system, producing safer and more useful responses.” <openai.com/product/gpt-4> Last Accessed 24 Apr. 2023.
- [29] Harshini, “Gpt-4 vs gpt-3.5: What is different?,” Mar. 2023. <www.analyticsinsight.net/gpt-4-vs-gpt-3-5-what-is-different> Last Accessed 24 Apr. 2023.
- [30] Google, “Using language to better interact with helper robots.” Website. <sites.research.google/palm-saycan> Last Accessed May 20 2023.
- [31] S. Narang and A. Chowdhery, “Pathways language model (palm): Scaling to 540 billion parameters for breakthrough performance.” Website, April 2022. <ai.googleblog.com/2022/04/pathways-language-model-palm-scaling-to.html> Last Accessed May 20 2023.
- [32] Y. Mehdi, “Confirmed: The New Bing Runs on OpenAI’s GPT-4.” Website, Mar 2023. <blogs.bing.com/search/march2023/Confirmed-the-new-Bing-runs-on-OpenAI's-GPT-4> Last Accessed May 20 2023.
- [33] D. Team, “Introducing duolingo max, a learning experience powered by gpt-4.” Website, March 2023. <blog.duolingo.com/duolingo-max> Last Accessed May 20 2023.
- [34] Salesforce, “Salesforce announces einstein gpt, the world’s first generative ai for crm.” Website, March 2023. <www.salesforce.com/news/press-releases/2023/03/07/einstein-generative-ai> Last Accessed May 20 2023.
- [35] O. Shulayeva, A. Siddharthan, and A. Wyner, “Recognizing cited facts and principles in legal judgments,” *Artif Intell Law* 25, p. 107–126, March 2017.
- [36] C. L. Giles, K. D. Bollacker, and S. Lawrence, “Citeseer: An automatic citation indexing system,” in *Proceedings of the Third ACM Conference on Digital Libraries*, NEC Research Institute, 1998.
- [37] A. Kanapala, S. Pal, and R. Pamula, “Text summarization from legal documents: a survey,” *Artif Intell Rev* 51, p. 371–402, 2019.
- [38] L. E. Resck, J. R. Ponciano, L. G. Nonato, and J. Poco, “LegalVis: Exploring and inferring precedent citations in legal documents,” *IEEE Transactions on Visualization and Computer Graphics*, vol. 29, pp. 3105–3120, jun 2023.
- [39] P. Zhang and L. Koppaka, “Semantics-based legal citation network,” (New York, NY, USA), Association for Computing Machinery, 2007.

- [40] C. Caragea, J. Wu, A. Ciobanu, K. Williams, J. Fernández-Ramírez, H.-H. Chen, Z. Wu, and L. Giles, “Citeseer^x: A scholarly big dataset,” in *Advances in Information Retrieval. ECIR 2014. Lecture Notes in Computer Science, vol 8416. Springer*, 2014. <citeseerx.ist.psu.edu>. Last accessed Jun. 4 2023.
- [41] World Wide Web Consortium (W3C), “Xml path language (xpath) 3.0,” April 2014. <www.w3.org/TR/xpath-30/>. Last accessed on June 13 2023.
- [42] M. Kay, “What’s new in 3.0 (xslt/xpath/xquery) (plus xml schema 1.1),” in *XML Prague 2012*, 2 2012. Retrieved on June 13 2023.
- [43] MDN Web Docs, “XPath.” <developer.mozilla.org/en-US/docs/Web/XPath>. Last accessed on June 13 2023.
- [44] “Postman.” <www.postman.com>. Last accessed June 13 2023.
- [45] “Natural Language Toolkit (NLTK) - Documentation.” <www.nltk.org/book/>. Last accessed on April 30 2023.
- [46] F. Neves, “Abreviaturas: lista de abreviações.” <www.dicio.com.br/abreviaturas-lista-de-abreviacoes>. Last accessed 4 June 2023.
- [47] NeuralMind AI, “portuguese-bert.” <github.com/neuralmind-ai/portuguese-bert>. Last accessed on June 18 2023.
- [48] Hugging Face, “neuralmind/bert-large-portuguese-cased.” <huggingface.co/neuralmind/bert-large-portuguese-cased>. Last accessed on June 18 2023.
- [49] Nishank Mishra, “An introduction to bert: How it works and how to use it for nlp.” <nish-19.github.io/posts/2021/02/blog-post-4>. Last accessed on June 20 2023.
- [50] NILC, “Mac-morpho corpus.” <nilc.icmc.usp.br/macmorpho/>. Last accessed on June 20 2023.
- [51] NILC, “Mac-morpho corpus manual.” <nilc.icmc.usp.br/macmorpho/macmorpho-manual.pdf>. Last accessed on June 20 2023.
- [52] R. Team, “Regex tutorial: Quantifiers.” <www.rexegg.com/regex-quantifiers.html>. Last accessed on June 20 2023.
- [53] Sinequa, “Classifying long textual documents up to 25,000 tokens using bert.” <sinequa.medium.com/classifying-long-textual-documents-up-to-25-000-tokens-using-bert-9d2dd55ca060>. Last accessed on June 20 2023.
- [54] P. Guillou, “NLP nas empresas | como ajustar um modelo de linguagem natural como BERT a um novo domínio linguístico com um adapter?,” *Medium*, Sept. 2021. Last accessed 25 Mar. 2023.

- [55] J. Brownlee, “Ensemble learning methods for deep learning neural networks.” Deep Learning Performance, December 2018.
- [56] W. J. Hutchins, “The history of machine translation in a nutshell,” 2006.
- [57] K. Reitz, “Requests-html: Html parsing for humans.” github.com/kennethreitz/requests-html. Last accessed on 13 June 2023.
- [58] F. Nogueira, F. Silva, S. Bugla, and M. R. Guimarães, “Identifying references to legal literature in portuguese superior court decisions - repository.” github.com/fabionoga/reference-identification. Last accessed on 30 June 2023.
- [59] A. Borodin, G. O. Roberts, J. S. Rosenthal, and P. Tsaparas, “Link analysis ranking: Algorithms, theory, and experiments,” February 2005.
- [60] A. Mowbray, P. Chung, and G. Greenleaf, “A free access, automated law citator with international scope: The lawcite project,” 12 2016.
- [61] S. N. Group, “Stanza – a python nlp package for many human languages.” stanfordnlp.github.io. Last accessed on June 26 2023.
- [62] C. Samarawickrama, M. de Almeida, N. de Silva, G. Ratnayaka, and A. S. Perera, “Party identification of legal documents using co-reference resolution and named entity recognition,” in *2020 IEEE 15th International Conference on Industrial and Information Systems (ICIIS)*, pp. 494–499, 2020.
- [63] N. Sannier, M. Adedjouma, M. Sabetzadeh, and L. Briand, “An automated framework for detection and resolution of cross references in legal texts,” in *Proceedings of the International Conference on Artificial Intelligence and Law*, 2021.
- [64] D. Bergmark, “Automatic extraction of reference linking information from online documents,” tech. rep., USA, November 2000.
- [65] CRACS & Inesc TEC, “Authenticus.” www.authenticus.pt. Last accessed on June 26 2023.
- [66] “Online confusion matrix.” onlineconfusionmatrix.com. Last accessed on June 26 2023.
- [67] E. C. Damasceno, “Business models in legal tech companies,” *Centro de Investigação Jurídica - FDUP*, August 2019.