Text2Icons: using AI to tell a story with icons

Joana Maria Lima Valente
Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos
Departamento de Ciência de Computadores
2021

Orientador
Alípio Mário Guedes Jorge, Professor Associado,
Faculdade de Ciências da Universidade do Porto

Coorientador
Sérgio Sobral Nunes, Professor Auxiliar,
Faculdade de Engenharia da Universidade do Porto

Todas as correções determinadas pelo júri, e só essas, foram efetuadas.

O Presidente do Júri,

Porto, ______/______/_________

MSC

Abstract

Narratives are used to convey information and are an important way of understanding the world through sharing that information. With the increasing development in Natural Language Processing (NLP) and Artificial Intelligence (AI), it becomes relevant to explore new techniques within these areas to extract, process, and visualize narratives. Narrative visualization tools can assist tasks such as text annotation, and can also allow a reader, for example of a news story, to have a different perspective from the traditional news format. This allows the news to be presented in a schematic way, with representative symbols or summarized with the relevant parts. With that in mind, we propose a new narrative visualization that uses icons to represent the elements that give narratives their meaning, thus giving a visual perspective of the news story. The approach we propose for the icon visualization is integrated in Brat2Viz, a Narrative Annotation Visualization tool that implements a pipeline that transforms annotations from text, into formal representations that lead to the construction of narrative visualizations, while using European Portuguese news *corpus* to demonstrate it. In order to build the icon visualization, we first present a narrative element extraction process that contains the automatic sentence extraction, the integration of automatic translation methods, and a generality level resolution algorithm that obtains the actors (narrative elements) most specific descriptions. Then, we introduce a method to create an icon dictionary to use as an icon database for the visualization, and a method to obtain and automatically search for icons, which altogether leads to the construction of this new graphic representation that uses icons. Furthermore, we present a critical analysis and evaluation of the generated results resorting to the responses collected by two surveys intended for this evaluation. Regarding the evaluation of the representation of the stories as a whole, 80% of the votes correspond to the generated visualization. As for the term-icon connection, in 85% of the questions, the icon generated by the visualization corresponds to the first or second most voted option. We also carried out evaluations related to the performance of the developed algorithm, the methods and icon sources that were integrated throughout the work.

Keywords: Narrative Visualization, Narrative Extraction, Icons, NLP, Embeddings

Resumo

As narrativas são usadas para transmitir informação, sendo uma forma importante de compreender o mundo através da partilha de informação. Com o crescente desenvolvimento em Processamento de Linguagem Natural e Inteligência Artificial, torna-se relevante explorar novas técnicas dentro dessas áreas para extrair, processar e visualizar narrativas. Ferramentas de visualização de narrativas tanto podem auxiliar tarefas como a anotação de textos, como podem permitir que um leitor, por exemplo de uma notícia, tenha uma perspetiva diferente do formato de notícia tradicional. Isto permite que a notícia seja apresentada de forma esquematizada, com símbolos representativos ou resumido com as partes relevantes. Com isso em vista, propomos uma nova visualização de narrativas que usa ícones para representar os elementos que dão sentido às narrativas, dando assim uma perspectiva visual da notícia. A abordagem que propomos para a visualização usando ícones está integrada no Brat2Viz, uma ferramenta de anotação e visualização de narrativas que implementa uma pipeline que transforma as anotações do texto em representações formais, que por sua vez levam à construção de visualizações de narrativas, utilizando um *corpus* de notícias de português europeu na demonstração. Para construir a visualização usando ícones, apresentamos primeiro um processo de extração de elementos narrativos que contém a extração automática de frases, a integração de métodos de tradução automática e um algoritmo de resolução de nível geral que obtém as descrições mais específicas dos atores (elementos narrativos). De seguida, apresentamos um método de criação de um dicionário de ícones para usar como base de dados de ícones na visualização e um método de obtenção e procura automática de ícones, que em conjunto levaram à construção desta nova representação gráfica que utiliza ícones. Além disso, apresentamos uma análise crítica e avaliamos os resultados gerados através das respostas recolhidas por dois inquéritos destinados a esta avaliação. Em relação à avaliação da representação das histórias como um todo, 80% dos votos correspondem à visualização gerada. Quanto à ligação termo-ícone, em 85% das perguntas, o ícone gerado pela visualização corresponde à primeira ou segunda opção mais votada. Também realizamos avaliações relacionadas com o desempenho do algoritmo desenvolvido, dos métodos e fontes de ícones que foram integrados ao longo do trabalho.

Palavras-chave: Visualização de Narrativas, Extração de Narrativas, Ícones, Processamento de Linguagem Natural, Embeddings

Acknowledgments

First of all, I would like to thank Professor Alípio and Professor Sérgio for their guidance during this work and for the opportunity to be part of this project. To the elements of the Text2Story project, I thank you for welcoming me, it was a pleasure to be part of your work.

Finally, my biggest thanks to my parents for the education they provided me, I am grateful to them and my brother, Alexandre, for always being on my side. Finally, I would like to thank João and my friends for all their support.

This research was carried out as part of the Text2Story project.
Text2Story is financed by the ERDF – European Regional Development Fund through the North Portugal Regional Operational Programme (NORTE 2020), under the PORTUGAL 2020 and by National Funds through the Portuguese funding agency, FCT - Fundação para a Ciência e a Tecnologia within project PTDC/CCI-COM/31857/2017 (NORTE-01-0145-FEDER-03185).
This funding fits under the research line of the Text2Story project.

Contents

List of Tables

List of Figures

Listings

Acronyms

API Application Programming Interface

AI Artificial Intelligence

ACE Automatic Content Extraction

BART Bidirectional and Auto-Regressive Transformers

BERT Bidirectional Encoder Representations from Transformers

Bi-LSTM Bidirectional LSTM

BRAT Brat Rapid Annotation Tool

BPE Byte-Pair Encoding

CharCNN Character-Level Convolutional Neural Network

CoVe Context Vectors

CBOW Continuous Bag-of-Words

CNN Convolutional Neural Network

DRS Discourse Representation Structure

DRT Discourse Representation Theory

DMCNN Dynamic Multi-pooling Convolutional Neural Network

EDU Elementary Discourse Unit

ELMo Embeddings from Language Models

FFNN Feedforward Neural Network

GRU Gated Recurrent Unit

Open AI GPT Generative Pre-trained Transformer

GCN Graph Convolutional Network

HNN Hybrid Neural Network

IE Information Extraction

IR Information Retrieval

JPEG Joint Photographic Experts Group

KE Keyword Extraction

LM Language Model

LIRICS Linguistic InfRastructure for Interoperable ResourCes and Systems

LSTM Long Short-Term Memory

MSC Message Sequence Chart

NER Named Entity Recognition

NLP Natural Language Processing

NLTK Natural Language Toolkit

NN Neural Network

NNLM Neural Network Language Model

POS Part-of-Speech

PNG Portable Network Graphics

PTM Pre-trained model

RAKE Rapid Automatic Keyword Extraction

RNN Recurrent Neural Network

SVG Scalable Vector Graphics

SRL Semantic Role Labeling

Stanford NER Stanford Named Entity Recognizer

TS Text Summarization

UML Unified Modeling Language

VoxML Visual Object Concept Modeling Language

YAKE! Yet Another Keyword Extractor

Chapter 1

Introduction

Narratives are designed to represent life events, conveying information, cultural values, teachings, and experiences, thus becoming an important way of perceiving the world through information sharing.

1.1 Motivation

Narrative extraction techniques are being developed to better understand the story behind texts, for example from news articles [87], social media [102], and medical reports [27]. Narrative representation has gained importance by facilitating tasks such as reading a news article by the reader [17], detecting a patient's diagnosis by a doctor [32], and performing financial analysis using financial narratives [35]. It is important to present narratives in new formats that can reach wider audiences, with a more useful, appealing and succinct language for the user. Narratives can be visualized in many different ways depending on the purpose and the audience. This work is motivated by the concept of exploring more appealing ways to represent narratives.

1.2 Text2Story Pipeline

This work was developed in the context of the INESC TEC partner project Text2Story[1]. This project aims to develop a conceptual framework and operational pipeline to extract narratives from textual sources. It is currently annotating a *corpus* of news stories in Portuguese, having developed a narrative annotation visualization tool [4] called Brat2Viz[2], which implements the pipeline that transforms annotations into formal representations (Discourse Representation Structure (DRS)), and then into visualizations.

[1] https://text2story.inesctec.pt/
[2] https://nabu.dcc.fc.up.pt/brat2viz

1.3 Objective

We present a graphical representation of a story using icons with the support of an automatic process that allows the extraction, processing and presentation of information. This graphical representation has as its starting point an initial text, on which the relevant narrative elements from the text are identified and extracted in order to produce an efficient representation to allow the reader to relate the icons to the story.

The main objectives of this thesis were achieved with the work developed and integrated as part of the Text2Story project pipeline. It is important to note that this work is designed for automatic extraction, but it is demonstrated with the set of annotated texts of the project.

1.4 Approach

To achieve our objective, we use the information regarding the extracted narrative elements stored in the DRS files, for our visualization. The graphical representation fits in the final step of our pipeline, as can be seen in Figure 1.1, this is where the icons that correspond to the story will be displayed. To generate the icon visualization by linking the narrative elements of news stories to icons that represent them, a connection was made between them, resorting to the construction of an icon dictionary. The dictionary was built to facilitate both the process of searching the most similar icon and saving them and their information. We have integrated several icon sources in the dictionary to aid the search, and made it to be extensible, i.e., to be able to automatically add new icons whenever necessary for new texts for which there is still no icon representation in the dictionary, thus allowing us to have a greater chance at reliably representing a new story.

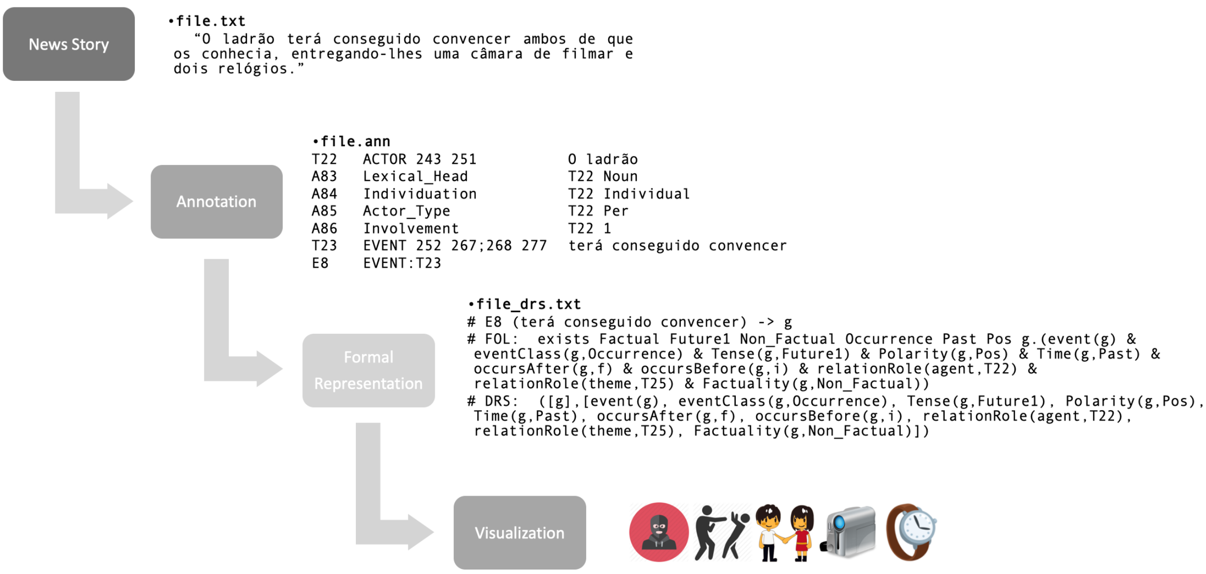


Figure 1.1: High-level pipeline steps.

1.5 Contributions

With the completion of this work, we obtained a method to automatically transform a story into a graphical representation using icons with little to none human intervention. Our contributions include:

1. A generality level resolution algorithm to get the most specific description from a set of descriptions, presenting its performance evaluation;
2. A new methodology to build dictionaries, not only for icons, but also for images, based on the integration of icon or image sources;
3. A method to automatically obtain icons:
    - Integration of two automatic translation methods, comparing their performance;
    - A semi-automatic icon search that allows the user to chose the icon to be saved;
    - An automatic search for the most similar icon to the searched term, by calculating the cosine similarity using word embeddings.
4. A new icon-based visualization method integrated in the pipeline, which can be seen in a vertical strip or in a slideshow:
    - Automatically extracting sentences from the texts;
    - An human evaluation of the proposed visualization solution.
5. Participation in IJUP[3] with the elaboration and presentation of an e-poster[4].

1.6 Document Layout

The thesis is organized as follows. The next two chapters introduce and explain concepts related to the topic under discussion. **Chapter 2** begins by introducing the concept of narrative and explaining and exemplifying some Natural Language Processing (NLP) tasks used for automatic extraction of narrative elements, also featuring neural language models and representation using embeddings. Followed by **Chapter 3**, where we explore some narrative visualizations. The icon resources explored for the visualization are presented in **Chapter 4**. In **Chapter 5** we describe the project pipeline, detailing the steps that constitute it and the visualizations already supported. Then, we present the methodology taken to extract narrative elements for our visualization, in **Chapter 6**. Followed by **Chapter 7**, where we describe the entire process of building the dictionary, its features and integration in the new visualization. Lastly, in **Chapter 8**, we present conclusions of this work and possible ideas for future work.

[3] https://ijup.up.pt/2021/
[4] Available at: https://ijup.up.pt/2021/wp-content/uploads/sites/657/2021/05/18645_Engineering.pdf

Chapter 2

Natural Language Processing

Natural Language Processing (NLP) is a theory-motivated range of computational techniques and area of Artificial Intelligence (AI) that comprehends all computer-based approaches to handle unrestricted written or spoken language. It aims to achieve automatic analysis and representation of the human language, by defining tasks that contribute to the conversion of text into a structured representation, thus deriving meaning from natural language input [9, 30].

Since narrative extraction aims to extract knowledge from natural language text to help the narrative building process, we present in this chapter the concept of narrative in Section 2.1, some NLP tasks used for automatic identification and extraction of narrative elements in Section 2.2, and cover in Section 2.3 neural language models and representation using embeddings. These NLP tasks lead to the understanding of natural language and allow us to analyze the text both syntactically and semantically [24], the syntax being the grammatical structure of the text, while the semantics is the meaning conveyed by the text. To analyze text information, we have tasks such as: Named Entity Recognition (NER), Semantic Role Labeling (SRL), Temporal Expression Recognition, Keyword Extraction (KE), Text Summarization (TS) and Event Extraction.

2.1 Narratives

The study of narratives is linked to several areas of study. Therefore, their understanding and definition has varied over time and depending on the author. According to Chatman [19], a narrative is defined as a structure with two parts: the story and the discourse. The story is the *what* that is portrayed in a narrative, while the discourse is the *how*. Prince [90], defines a narrative as the recounting of one or more events, with a continuant subject and constitute a whole. Riedl et al. [99], explains narratives as cognitive structures designed to represent life events to better understand the world around us.

In short, what makes up a narrative are the entities, the temporal data, and the events found in it, thus giving it meaning.

2.2 Automatic Extraction of Narrative Elements

Tasks covered by NLP enable the conversion of unstructured text into a structured form, and therefore allow automatic identification and extraction of information from narratives. For each task covered here, we provide an overview with representative examples and the current state-of-the-art tools and models or systems.

For example purposes of the NLP tasks covered in this section, consider the following sentence as a running example (1):

(1) John and his friends robbed three stores this Monday morning in Lisbon. One of them wielded a shotgun to threaten employees and customers. The burglars fled on foot with the money. The police went to the scene.

2.2.1 Named Entity Recognition

NER is a NLP task that aims to identify and categorize named entities in unstructured texts, assigning them labels to pre-defined categories such as "PERSON", "DATE" or "LOCATION", thereby helping to answer questions about the text [24]. A **named entity** is a "real-world object" that is assigned a name of a category, such as the names of companies, objects, persons, locations, organizations and products. The "named entity" task was introduced at the Sixth Message Understanding Conference (MUC-6) [48], with the aim of identifying the names of people, organizations and geographic locations in a text.

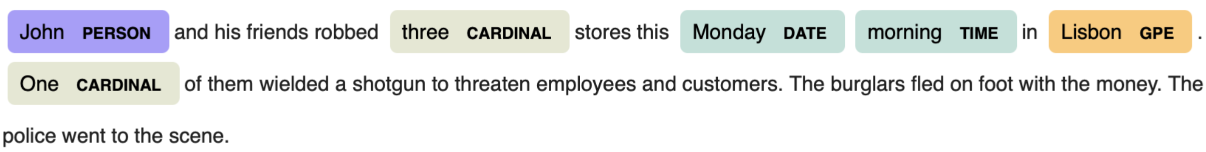


Figure 2.1: Result from applying NER to our running example.

Figure 2.1 shows the result of applying NER to our running example (1), using displacy[1] from spaCy[2]. The following entities are identified and labeled:

"John": PERSON People, including fictional;
"three" and "One": CARDINAL Numerals that do not fall under another type;
"Monday": DATE Absolute or relative dates or periods;
"morning": TIME Temporal units smaller than a day;
"Lisbon": GPE Countries, cities, states.

[1] https://spacy.io/usage/visualizers#ent
[2] https://spacy.io

Below, some of the available tools capable of performing NER are briefly presented:

- **spaCy**: a Python open-source software library for advanced NLP, with models trained on more than one language. It currently has models for 15 languages and one multi-language model with 10 languages. This is useful for NLP tasks such as NER;
- **Stanford CoreNLP**[3]: Java toolkit for NLP tasks developed by Stanford University that currently supports 6 languages. Stanford Named Entity Recognizer (Stanford NER)[4] is an implementation of a Named Entity Recognizer that is part of Stanford CoreNLP, which can be seen as a smaller tool for a specific task;
- **Natural Language Toolkit (NLTK)**[5]: platform of libraries for Python widely used for NLP tasks. NLTK has its own classifier to recognize named entities, but also provides a module[6] for interfacing with the Stanford NER tagger.

NLP task models or systems are usually evaluated using precision, recall and F_1 measure. For the NER task, we call **precision** to the percentage of named entities correctly found by the system. The percentage of named entities present that are found by the system is called **recall**. Namely, a correct named entity is an exact match with the corresponding entity in the data file [122]. The F_1 **measure** is a measure of a test's accuracy that is calculated from the values of precision and recall.

2.2.2 Semantic Role Labeling

SRL is the task of automatically finding the semantic roles of each argument of each predicate in a sentence, i.e., the identification and labeling of arguments present in a text. This NLP task is focused on characterizing events through semantic analysis of sentences, defining "who" did "what" to "whom", "where", "when", and "how", this way identifying participants and properties of the events [60, 74, 110].

Given a phrase and a designated verb, the SRL task consists of argument identification and classification. The argument identification aims to identify the boundaries of the arguments of the verb predicate, while the classification labels them with semantic roles [74].

The SRL task can be divided in two main approaches: span-based SRL and dependency-based SRL. The first is based on constituency-based parsing, aiming to find the word spans, i.e., groups of contiguous words that compose arguments of a verb and correctly label them. The second, is based on dependency-based parsing in order to find only the head word of the argument, instead of the whole argument [110].

[3] https://stanfordnlp.github.io/CoreNLP/
[4] https://nlp.stanford.edu/software/CRF-NER.shtml
[5] https://www.nltk.org
[6] https://www.nltk.org/_modules/nltk/tag/stanford.html

Illustrated in Figure 2.2, we can see SRL performed to the top of the first sentence of our running example (1), using AllenNLP[7] model reimplementation of a Bidirectional Encoder Representations from Transformers (BERT)-based model [106].

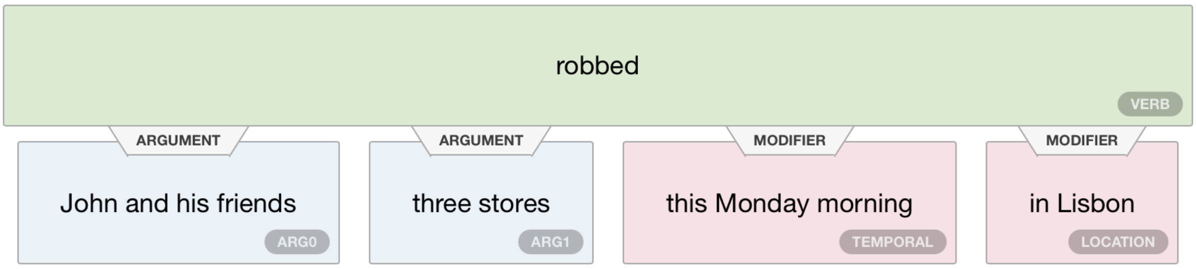


Figure 2.2: SRL applied to a portion of our running example.

Here are some useful tools to perform SRL:

- **PropBank**[8]: Proposition Bank or PropBank, is the most widespread resource for training and evaluating systems for the English language. Provides predicate-argument annotation for the entire Penn Treebank [73] and contains information about the location of the verb, and the location and identity of its arguments. Propbank.Br [33] is the Brazilian Portuguese version of PropBank, following the same annotation style;
- **VerbNet** [101]: verb lexicon compatible with WordNet[9] which is a large English lexical database, but with explicitly stated syntactic and semantic information. It organizes verbs into classes according to which semantic roles the verbs allow;
- **nlpnet**[10]: Python library for NLP tasks based on neural networks. It supports English and has some functions that were specifically designed to work with Brazilian Portuguese [41].

In SRL, the **precision** is the proportion of predicted arguments that are correct and the **recall** is the proportion of true arguments predicted by the system. Both are calculated for each role being predicted, as well as for the total predictions from a model or system. The common data sets to evaluate span-based SRL are the data sets from CoNLL-2004 [15], CoNLL-2005 [16] and CoNLL-2012 [89] shared tasks. As for dependency-based SRL, the common data sets are the CoNLL-2008 [120] and CoNLL-2009 [50]. All these data sets are based on data sets from the PropBank project [110].

[7] https://demo.allennlp.org/semantic-role-labeling/
[8] http://www.nltk.org/howto/propbank.html
[9] https://wordnet.princeton.edu
[10] https://pypi.org/project/nlpnet/

2.2.3 Temporal Expression Recognition

Temporal annotation of documents with the extraction and chronological ordering of events, becomes important in many NLP applications. Temporal information regarding a specific point in time, can be expressed in many different ways and most of that temporal information is often left implicit in a text with many vague temporal expressions. It is therefore crucial for automatic extraction of information from texts to have automatic question answering or summarization applications with the ability to identify what events are being described and when these events occurred [93, 113].

This task of temporal tagging can be split into two pre-processing subtasks: extraction and normalization. The **extraction** subtask focus on correctly identifying temporal expressions and their boundaries. **Normalization** is a more complex subtask, since it is responsible for the correct attribution of a value to temporal expressions in a standard format, following a standard ISO for temporal annotations [115].

According to Strötgen and Gertz [115], temporal expressions can be distinguish between: explicit, when fully specified, like specific dates (e.g., "September 18, 2002"); implicit, can be normalized once their temporal semantics is known, such as names of holidays (e.g., "Christmas 2019"); and relative expressions that need more context information to be normalized (e.g., "today").

Figure 2.3, shows the extracted temporal expressions in our running example (1), marked in blue using HeidelTime online demo[11]. Note that the instantiated date corresponds to the date of execution of the experiment.

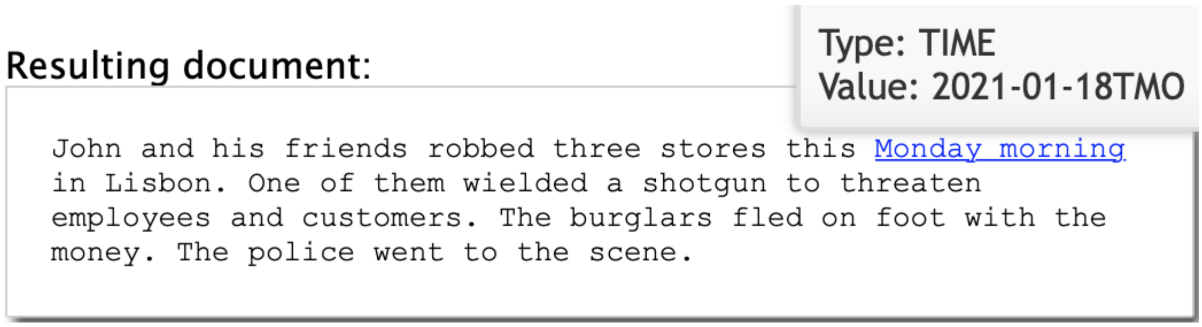


Figure 2.3: Recognizing temporal expressions in our running example.

Research in this field has led to the emergence of markup languages like TimeML [93], annotated corpora such as TimeBank *corpus* [94] and competitions, for example TempEval challenges [7].

The markup annotation language, TimeML[12], was presented in an attempt to capture temporal and event related information in language, becoming the ISO standard for temporal

[11] https://heideltime.ifi.uni-heidelberg.de/heideltime/
[12] http://www.timeml.org

annotation. It has four different tag types to annotate events, times and temporal relations. The TIMEX3 tag is used to capture all temporal expressions, capturing four different types: TIME, DATE, DURATION, and SET, the EVENT tag for all temporal events, the SIGNAL tag to annotate functional words, and LINK tag to represent all relationships between the other tags.

TimeBank *corpus* was created as a reference *corpus* for TimeML. It contains TIMEX3 tags for temporal expressions, annotating events and temporal relations. The 1.2 version [94], consists of 183 news articles annotated following TimeML 1.2.1[13].

TempEval challenges [126] aim to identify temporal relations in text. TempEval is an evaluation initiative that serves to evaluate systems that automatically annotate texts with temporal relations. It was created for the SemEval 2007 workshop [1] for temporal relation evaluation using the TimeML annotation language.

When it comes to temporal taggers, we have Heideltime [113], SuTime [18] and GuTime [71], leading the state-of-the-art.

- **HeidelTime**[14]: Strötgen and Gertz introduced HeidelTime in 2010, a system for the extraction and normalization of temporal expressions in English documents. It was the best-performing system in Task A for English in TempEval-2 [127]. In 2012, they integrated strategies for temporal tagging in different domains [114], since a temporal tagger should be aware of the domain associated with the documents being processed and therefore apply domain-specific strategies. In 2014, Chinese temporal tagging [69] was added to its resources. In the same year, it was extended to extract and normalize temporal expressions referring to historic dates [118], and resources were developed for Arabic, Vietnamese, Spanish and Italian [117]. With temporal taggers usually being developed separately for each language, in 2015, they presented an approach to extend HeidelTime to all languages [116];

- **SuTime**[15]: Java library for recognizing and normalizing time expressions. It is a rule-based system particularly devoted to the English language, available as an annotator that is part of the Stanford CoreNLP[16] pipeline, and can be used to annotate documents with temporal information. These annotations are in the form of TIMEX3 tags that are part of TimeML [93];

- **GuTime**[17]: Temporal expression tagger in Perl[18] created by Georgetown University, part of the TARSQI Toolkit [125] that relies on rules to recognize and normalize values of temporal expressions in English. Like HeidelTime and SuTime, GuTime also supports the four basic types of temporal objects defined in TimeML.

[13] http://www.timeml.org/publications/timeMLdocs/timeml_1.2.1.html
[14] https://github.com/HeidelTime/heideltime
[15] https://nlp.stanford.edu/software/sutime.shtml
[16] https://stanfordnlp.github.io/CoreNLP/
[17] http://www.timeml.org/tarsqi/modules/gutime/index.html
[18] https://www.perl.org

2.2.4 Keyword Extraction

Keywords are often provided about the content of documents just like newspapers do for their articles. Some authors prefer the term keyphrases over keywords because sometimes they consist of more than one word. So to describe the most important contents of a text, keywords or keyphrases are used, with different approaches to select them.

Keywords are represented in a condensed form, providing a compact representation of a document's content. Being widely used for analysis, indexing and retrieval of information. KE connects different areas like Information Retrieval (IR) and NLP. It can be defined as the process that automatically identifies a set of elements that best describes the subject of a text by providing metadata that summarizes and characterizes the document. In large document collections, like digital libraries, it is valuable to have a set of keywords that describe them. With the extremely large number of documents available online, manual KE is not doable. So KE programs are frequently used to collect the main idea of a document and create or select keywords. Therefore, research has been done to automatically extract keywords from documents [13, 119, 123, 128].

There are two types of approaches: unsupervised and supervised. The **supervised**, has prior knowledge of the expected results, while the **unsupervised**, tries to learn without prior knowledge [111]. Most traditional approaches are supervised, making it dependent on access to training annotated text corpora. Even though it is more efficient, its training process is relatively long [11].

Illustrated in Figure 2.4 are the extracted keywords of our running example (1) highlighted in white, using Yet Another Keyword Extractor (YAKE!) Demo[19], an unsupervised approach.

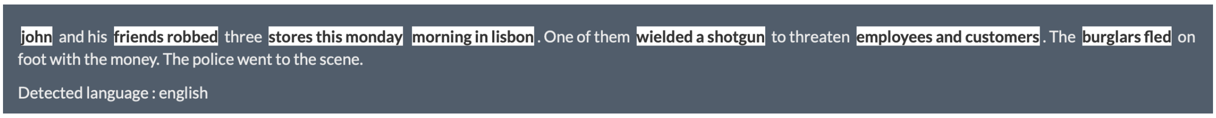


Figure 2.4: Extracting keywords from our running example.

The increased need for this task to be done properly and efficiently, led to the emergence of automatic keyword extraction tools, such as those presented below:

- **Rapid Automatic Keyword Extraction (RAKE)** [100, 119]: unsupervised, extremely efficient, domain-independent, and language-independent method to extract keywords from individual documents to enable application to dynamic collections. It is easily applied to new domains, and performs well on several types of documents, specially those that do not follow specific grammar conventions. RAKE has as input parameters a list of stop words, a set of phrase delimiters, and a set of word delimiters, so the document is partitioned into candidate keywords. It does not require a training set;

[19] https://yake.inesctec.pt

- **YAKE!** [11–13]: automatic keyword extraction using text features with an online keyword extraction demo[20] and a Python implementation available at PyPi[21]. This is an unsupervised approach that builds upon statistical features extracted from text to identify and rank the most important keywords. It is adaptable to documents written in many languages without needing outside knowledge other than a static list of stopwords. It does not need to be trained on a particular set of documents and does not use NER or Part-of-Speech (POS) taggers. Therefore, it can be easily applied to single texts, regardless of the existence of a *corpus*, dictionary or any external collection;

- **KeyBERT** [105]: self-supervised technique of keywords and keyphrases retrieval and extraction that uses BERT, and can perform self labeling of unlabeled *corpus*. BERT [28] is a language representation model that transforms phrases and documents to vectors that collect their meaning. KeyBERT[22] is a BERT-based solution that does not have to be trained from scratch.

2.2.5 Text Summarization

TS aims to summarize relevant and complementary information in narratives. Automatic TS focuses on identifying important content, usually at sentence level. A summarization system can extract the important sentences identified to form an output summary. The methods used for TS are known as either extractive or abstractive [31, 79, 133].

The **extractive** identifies the most appropriate words or sentences by selecting salient snippets or passages from the input document, and concatenates them to form a summary. Works that use this methods have two main steps: sentence scoring (to assign an importance score to each sentence) and sentence selection (to choose content sentence by sentence).

The **abstractive**, on the other hand, generates summaries and produces novel words and sentences to concisely paraphrase the information in the documents.

In comparison, abstract algorithms are more flexible and more likely to produce fluent and coherent summaries than extractive ones. However, extractive methods for summarization have been proven in many systems, dominating the summarization research.

Summarization models differ in the types of architectures (with Encoders: Convolutional Neural Network (CNN) [63], Recurrent Neural Network (RNN) [21], Long Short-Term Memory (LSTM) [55], Transformer [124]; and Decoders: auto-regressive, non auto-regressive), external transferable knowledge (GloVe [84], BERT [28], NEWSROOM [49]), and learning schemas (supervised learning and reinforcement learning) [132].

The summarization of our running example (1) is presented in Figure 2.5 using SMRZR.io[23],

[20] https://yake.inesctec.pt
[21] https://pypi.org/project/yake/
[22] https://github.com/MaartenGr/KeyBERT/
[23] https://smrzr.io

an online BERT-based summarization tool [78].

Text reduced by **68%** (37 to 12 words)

John and his friends robbed three stores this Monday morning in Lisbon.

Figure 2.5: Summarizing our running example.

Some methods and techniques to perform TS are presented below:

- **NeuSum** [133]: the Neural Extractive Document Summarization is a framework that learns together both to score and to select sentences, integrating them into one end-to-end trainable model. Becoming a hierarchical document encoder, and a sentence extractor. It is a neural network model that learns to identify the relative importance of sentences, predicting the relative gain given the sentence extraction and the partial output summary. This model has two parts, the document encoder and the sentence extractor;
- **DiscoBERT** [130]: discourse-aware neural summarization BERT-base model that extracts discourse units, instead of sentences, as candidates for extractive selection. It performs compression with extraction simultaneously and reduces redundancy across sentences. The model consists of a **document encoder** that uses a pre-trained BERT model to encode the document, an **extractor** to obtain Elementary Discourse Unit (EDU) representations, and a **graph encoder** to update the EDU representations;
- **GSum** [31]: extensible guided summarization framework that can take different kinds of external guidance[24] as input. The model is based on neural encoder-decoders with pre-trained language models, like BERT [28] and Bidirectional and Auto-Regressive Transformers (BART) [68].

2.2.6 Event Extraction

An **event** is defined in the Automatic Content Extraction (ACE) evaluation [29] as a specific occurrence with one or multiple participants, being each participant characterized by a role played in the event. That said, Event Extraction is the process of obtaining knowledge about something significant that happens at a specific time and place [2], automatically identifying information about what happened and when it happened.

Event detection and extraction is a recurrent, important and challenging task in Information Extraction (IE) and NLP with most state-of-the-art methods treating it as a classification task

[24] Guidance can be defined as some variety of signal that is fed into the model in addition to the source document, provided in different forms, including tokens, triples, sentences and summaries [31].

using features extracted by textual analysis. These features can be divided into two categories: lexical features and contextual features. The **lexical features** aim to capture semantics of words, containing POS tagging, entity information, and morphology features like token and lemma. A **token** is defined as an instance of a sequence of characters in a document, which are grouped as a semantic unit for processing, and a **lemma** is the base form of a word [72]. On the other hand, **contextual features** aim to comprehend how facts are connected. These feature-based methods are known to be effective but costly in terms of the work required, needing, for example, manually designed feature templates and expert knowledge [20, 38].

ACE defines an **event mention** as a sentence where an event is described, an **event trigger** as the word that expresses the occurrence of an event, an **event argument** as an entity mention, a temporal expression or a value involved in an event, and an **argument role** as the relationship between the argument and the event in which it participates. Event detection and extraction using our running example (1) is illustrated in Figure 2.6, using the ACE 2005's English Annotation Guidelines for Events [25]. In the figure, on the left, is the predefined event schema and on the right, below our running example, the type of event, the event trigger, and the arguments are identified.

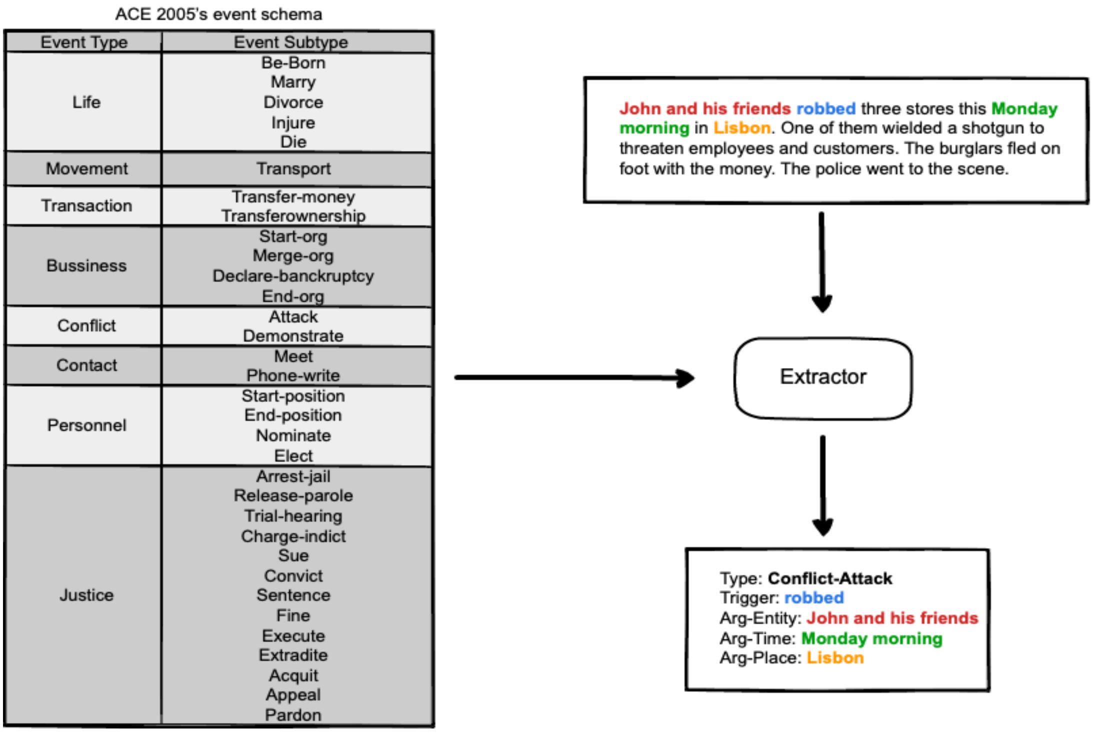


Figure 2.6: Result of manual Event Extraction for our running example.

Some works related to event detection models are presented below:

- **Dynamic Multi-pooling Convolutional Neural Network (DMCNN)** [20]: a framework for event extraction, which can automatically induce lexical-level and sentence-level features from plain texts without elaborated NLP pre-processing. With a DMCNN the aim is to capture valuable information in a sentence for event extraction. They describe the event extraction as a two-stage classification. The first is the trigger classification that uses a DMCNN to classify each word in a sentence to identify trigger words. If a sentence with triggers is found, they proceed to the second stage, the argument classification, applying a similar DMCNN to assign arguments to triggers and align the roles of the arguments;

- **Hybrid Neural Network (HNN) model** [37, 38]: language-independent neural network approach with focus on learning a representation of each word in a sentence, in order to predict whether the word is an event trigger or not. It combines two types of neural networks: a Bidirectional LSTM (Bi-LSTM) used to encode the semantics of each word with its preceding and following information, and a CNN to capture the structure information from local contexts. This model is language independent without any supervised NLP tools and resources;

- **Timeline Extraction Task** [88]: a tool that extracts structured information of events from a stream of online news articles, for a given entity, currently only for English. It forms a target-entity event timeline because it also extracts entity related relations linking and classifying them temporally. The task consists of extracting a list of time-ordered tuples, given a set of documents and a target entity of interest. For this task they define an event as any situation that involves the target entity for a period of time.

2.3 Neural Language Models and Embeddings

2.3.1 Neural Language Models

Language Models (LMs) are a core component of NLP, being the basis of several NLP tasks, such as translation tasks, speech recognition, and text classification, by evaluating probabilities of word sequences, and providing word representations that give context that allows to distinguish similar words and phrases. Neural Network Language Models (NNLMs) [56] emerged as an improvement over traditional LMs, both in terms of dimensionality and performance. Before NNLMs appeared, the n-gram model, an approximation method, was the most widely used state-of-the-art model. An **n-gram** is a contiguous sequence of n words, for example, a 2-gram is a bigram, i.e., a two-word sequence of words [58].

Neural Networks (NNs) [131] are learning models widely used in several areas of computer science. Some of the NNs introduced for language modeling in continuous space included Feedforward Neural Networks (FFNNs) and RNNs, which are capable of automatically learning features and continuous representations [5, 46].

Pre-training is an effective way to learn parameters of deep neural networks. Pre-trained models (PTMs) [95] applied to large *corpus* can learn universal language representations, which benefits NLP tasks and can prevent the need to train new models from scratch. The first PTMs to appear, called **first-generation** PTMs, are pre-trained word embeddings, which will be explored in the next subsection, such as Skip-Gram [76] and GloVe [84], aimed to learning good word embeddings. These embeddings capture semantic meanings of words, are context-free, and miss to capture higher-level concepts within context, like syntactic structures and semantic roles. The **second-generation** of PTMs, the pre-trained Contextual Encoders, are focused on learning contextual word embeddings, also known as the output vectors of neural encoders since they represent the word semantics depending on the context. These encoders include, among others, Context Vectors (CoVe) [75], Embeddings from Language Models (ELMo) representations [85], Generative Pre-trained Transformer (Open AI GPT) [96, 124], and BERT [28].

As pointed out by Qiu et al. [95], the advantages of pre-training include: when pre-training in large text *corpus*, it can lead to the learning of universal language representations, helping with downstream tasks like NER; gives a better initialization to the model, leading to better performance and speeding up the intended task; and, shown by Erhan et al. [36], pre-training acts as a form of regularization to avoid overfitting in small data.

2.3.2 Embeddings

Representation learning means learning representations of data in order to facilitate the extraction of important information when building classifiers or predictors. A good representation is one that, according to Bengio et al. [6], identifies the elements that explain the observed input, being useful as an input for a supervised predictor. With regards to language representation learning, it is considered a good representation when implicit linguistic rules and common knowledge present in the text are identified and obtained, such as lexical meanings, syntactic structures and semantic roles [95].

Embeddings are vector representations, and in NLP the term **word embedding** is used in word representation when performing text analysis. This representation is presented in the form of a vector that, when encoding the meaning of the word, makes words with similar meanings close to each other in the vector space. These representations are used in all NLP applications that resort to the meaning of words [59]. Word embeddings are obtained by mapping words or sentences to vectors, using modeling languages and feature learning techniques. An important focus of research in NLP is to automatically learn useful representations from input text [6].

Word embeddings can be of two types depending on whether the embedding for a given word dynamically changes depending on the context in which it occurs, so it can be either non-contextual or contextual embedding [80, 95]:

- **Non-contextual Embeddings:** these are static embeddings. The word static here means that the embedding for a word will always be the same regardless of context, i.e., it will not be able to encode polysemous words. These embeddings are still unable to handle unusual vocabulary. Some models capable of dealing with these problems emerged by proposing character-level word representations or subword representations, which are widely used in NLP tasks, such as Character-Level Convolutional Neural Network (CharCNN) [64], FastText [8] and Byte-Pair Encoding (BPE) [103];

- **Contextual Embeddings:** embeddings that address the problem of polysemous words, and the fact that the nature of words depends on their context, in order to distinguish the semantics of words in different contexts, by taking into account their contextual information. For this, the contextual representation of words and subwords depends on the entire text in which they are inserted. These neural contextual encoders can be of two categories: sequence models (LSTMs [55, 121] and Gated Recurrent Units (GRUs) [21]) and non-sequence models (Recursive NN [109], TreeLSTM [134], and Graph Convolutional Network (GCN) [65]).

Several related resources to word representations include open-source implementations, like pre-trained word vectors and libraries, which are available online. The following list summarizes some of them:

- **word2vec**[25] [76]: technique for learning word embeddings that provides a tool[26] with implementations of the Continuous Bag-of-Words (CBOW) and the Skip-Gram model, in order to make pre-trained word embeddings accessible for different NLP tasks;
- **fastText**[27] [8, 77]: library for learning word representations and sentence classification that provides several pre-trained word vectors[28] for 157 languages. It considers subword information, and represents words by a sum of its character n-grams;
- **Transformers**[29] [129]: library with tools and access to pre-trained models, providing several models[30]. It has JAX[31], PyTorch[32] and TensorFlow[33] integrated in its framework.

2.4 Summary

In this chapter, an overview of NLP concepts and techniques is presented. Information related to narratives was covered, processes of extracting narrative elements were explained, and neural language models and representation using embeddings were addressed. It is now important to understand how narratives can be represented, how techniques of automatic extraction of narrative elements can be used to obtain information about a story, and to understand the value of representation of information using embeddings.

[25] https://github.com/tmikolov/word2vec
[26] https://code.google.com/archive/p/word2vec/
[27] https://github.com/facebookresearch/fastText
[28] https://fasttext.cc/docs/en/crawl-vectors.html
[29] https://github.com/huggingface/transformers
[30] https://huggingface.co/models
[31] https://jax.readthedocs.io/en/latest/
[32] https://pytorch.org
[33] https://www.tensorflow.org

Chapter 3

Narrative Visualization

Stories are used, in addition to transmitting information, to convey cultural values, teachings and experiences, and can appear in different forms: books, illustrations, news, movies, theaters, music, and dances. Narratives are an important way of understanding the world through information sharing, which raises an important issue: the amount and complexity of information available. Knowledge and information visualizations are widely applied in areas such as education, in order to help users process, access, and handle complex knowledge and vast amounts of information [62]. The importance and potential of storytelling has been researched for information visualization as an efficient method of data representation, such as the work presented by Figueiras [39] that addresses the benefits of incorporating narrative elements in visualizations.

There has been a lot of research focused on this topic. This can be seen in articles such as those presented by: Palshikar et al. [81] where they present the extraction of Message Sequence Charts (MSCs) from historical narratives; Shamsi et al. [104] propose a schema based on emojis and search engines, presenting a graphics-based approach for information seeking and retrieval; Pustejovsky et al. [92] present Visual Object Concept Modeling Language (VoxML), a modeling language for constructing 3D visualizations from concepts denoted by natural language expressions; and Ramesh et al. [97] describe a text-to-image generation approach[1], featuring a neural network called DALL·E[2] that is trained to generate images from text captions for a wide range of concepts expressible in natural language, using a dataset of text–image pairs.

This chapter focuses on alternative approaches of narrative visualization, some of which are explored here. In Section 3.1, a brief summary of infographics is presented. And then, in Section 3.2, we will see timelines as temporal representations of information.

[1] Code available at: https://github.com/openai/dall-e

[2] https://openai.com/blog/dall-e/

3.1 Infographics

Information graphics, also known as infographics, are used to represent complex data to communicate information, data or knowledge. This is done by joining artistic elements of data, such as images, with content derived from data like graphics, and with texts. Infographics have become an increasingly popular way to spread data, with the growing addition of interactive elements to online infographics, which facilitates a more detailed and complex storytelling by the journalists, and an easier understanding while captivating the reader [26, 52, 67].

The infographics section[3] of Jornal Público has a project [3] that uses infographics for storytelling using iconography as a way of visualizing information, simplifying and facilitating the reader's understanding. They develop infographics for both digital and print platforms, creating both infographics that support an article and autonomous infographics articles, as the one shown in Figure 3.1.

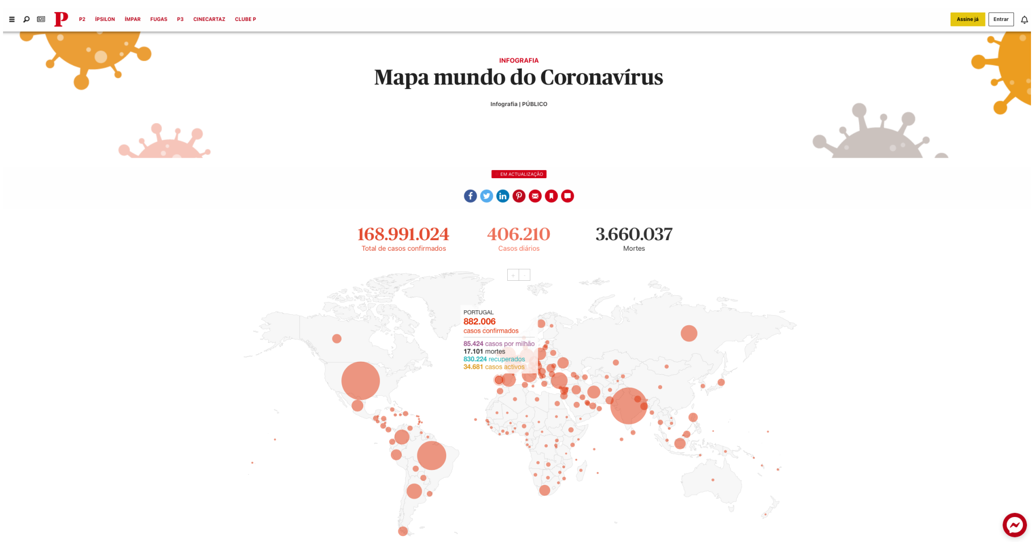


Figure 3.1: Example of an autonomous infographic article taken from Jornal Público online edition.

In a path similar to the work done by Público, the Graphical Storytelling project[4] is researching ways to generate short news comics based on journalistic text in order to tell stories in an accessible and appealing way.

3.2 Timelines

Information Retrieval (IR) is an area where methods and approaches are developed, offering users the most relevant documents from an existing collection, having been successful in providing

[3] https://www.publico.pt/multimedia/infografias

[4] https://bbcnewslabs.co.uk/projects/graphical-storytelling/

access to the vast amount of information available online [10, 61].

The growing amount of online information leads to the development of new tools for those who want to understand event details without needing to read documents in their entirety. Solutions include extracting, summarizing and organizing information to help understand how events are related. The use of timelines to support storytelling as a way to organize complex events is a common approach that uses information spread over multiple documents in a temporal order, and a solution used by media outlets, despite its manual construction being laborious, time-consuming and requiring resources capable of perceiving natural language. Therefore, tools have arise to present text using temporal structures, thus reducing the reader's effort [14, 82]. Analyzed below, Time-Matters and Conta-me Histórias are two of the solutions that emerged.

3.2.1 Time-Matters

Time-Matters [14] is a system that aims to score the relevance of temporal expressions that can be found in a text, whether in a single document or in a set of documents. Providing users with an automatic view of important periods in time, and associated text stories without taking so much time as there is no need to read extensive documents.

The purpose of this system is to estimate the importance of temporal expressions in a text and then discard those that have no meaning. In Figure 3.2a we can see exactly this in the annotated text with the temporal expressions presented in a relevance scale. The system offers the chance to interact with it through the storyline temporal element, as can be seen in Figure 3.2b. Having as key points the fact that it is an unsupervised approach, it is independent of domain and *corpus* as it does not need a training step, and is built from the characteristics of any text, as well as being language-independent. It uses the previously mentioned temporal tagger, HeidelTime [115], to detect temporal expressions.

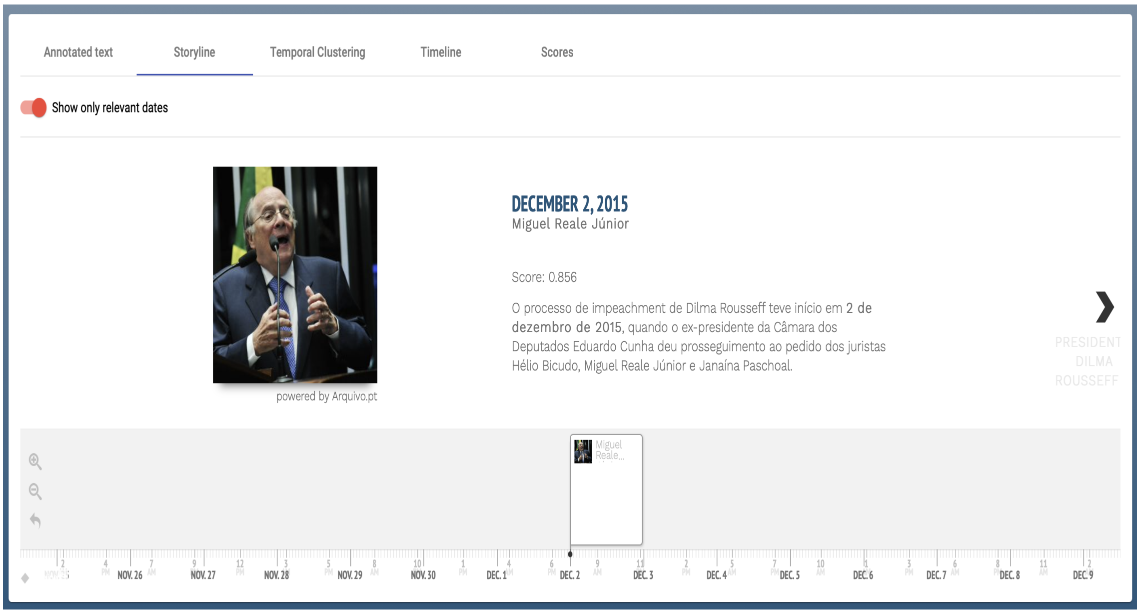
The authors made available contributions from Time-Matters, including an online demo[5], an Application Programming Interface (API)[6], a Python package[7] and a Docker image[8].

3.2.2 Conta-me Histórias

Conta-me Histórias [82] (Tell me stories) is a tool that automatically generates a temporal summarization of news collections. It is available in two versions, one in Portuguese and one in English, each having its name in the corresponding language. Through *Conta-me Histórias*, the Portuguese online demo[9], it allows users to interact with past events by exploring the Portuguese Web Archive [47] with news from 24 Portuguese media outlets. The English version, *Tell me*

[5] http://time-matters.inesctec.pt
[6] http://time-matters.inesctec.pt/api
[7] https://github.com/LIAAD/Time-Matters
[8] https://hub.docker.com/r/liaad/time-matters
[9] http://contamehistorias.pt/arquivopt/

(a) Annotated Text.



(b) Storyline.

Figure 3.2: Illustrative example taken from Time-Matters online demo, with the annotated text (a) and the respective storyline (b).

stories[10], uses Bing News Search API[11] to create narratives. An illustrative example of Conta-me Histórias can be seen in Figure 3.3.

This temporal summarization framework (its code is available online[12]) consists of: **news retrieval** of data of interest, **term weighting** by calculating each term weight with an adaptation

[10] http://nlplab.inesctec.pt/tellmestories/
[11] https://www.microsoft.com/en-us/bing/apis/bing-news-search-api
[12] https://github.com/LIAAD/TemporalSummarizationFramework

Figure 3.3: Illustrative example taken from Conta-me Histórias online demo.

of Yet Another Keyword Extractor (YAKE!) [11–13] keyword extractor, **identifying relevant time intervals** by identifying main events, **computing headline scores** to determine the most important headlines for each temporal interval, and **de-duplication** to eliminate similar key-phrases.

This tool is available on Google Play[13], and the authors also offer an engine as an open source package that can be extended to support different datasets or languages.

[13] https://play.google.com/store/apps/details?id=com.app.projetofinal

3.3 Summary

In this chapter, we presented various types of visualizations applied to different kinds of information. Infographics represent information in a more artistic way, becoming appealing in journalistic contexts, as it can be useful to use as a way to captivate new audiences. Timelines, on the other hand, focus on the extraction and representation of temporal events, making timelines summarized and dynamic for the user. Journalists can be the consumers of these timelines, using them as a working tool to summarize and deal with news articles.

Now with a deeper understanding of the importance of using visualizations to represent information, and with a broader view of its use in different contexts and with different purposes, we focus our research on resources related to the icon visualization.

Chapter 4

Icon Resources

According to the online Cambridge Dictionary[1], the Oxford Learner's Dictionaries[2], and Merriam-Webster[3], an icon is a small graphic picture or symbol on a computer screen that can be pointed at and clicked on that can represent a program or a file. It can also be seen as a sign whose form suggests its meaning.

Icons are important components in the interaction between humans and computers. With increasing technological development, icons have been optimized depending on their purpose. As a result, increasingly large numbers of icon design styles and interfaces have emerged [70]. This graphical symbol can be used to represent a wide range of things, from an application or a folder on a computer, as already mentioned, to a command or button that performs an action, but above all to represent something.

Within the scope of this work, icons are used to represent elements of a narrative, thus giving a visual perspective of a narrative. Keeping in mind that the main objective is to achieve a graphical representation of a story using icons, the construction of an icon dictionary was outlined to serve as an icon database for the story visualization. A challenge we encountered at this stage of the work was to define how to obtain the icons to compose the dictionary, thus making it relevant to present the research carried out on the use of icons and some possible sources where they can be obtained from to use in the visualization.

In this chapter, the sources that have been researched and explored to analyze whether they can be used as a source of icons for the dictionary are described. These include icon and emojis Application Programming Interfaces (APIs), datasets, and perhaps the use of unicode characters. An evaluation and a summary of the characteristics of the sources explored are also presented at the end of this chapter.

[1] https://dictionary.cambridge.org
[2] https://www.oxfordlearnersdictionaries.com
[3] https://www.merriam-webster.com

4.1 emojidex

The open source project emojidex[4] consists of a set of tools and assets that provides emojis as a service, with a growing collection of over 3,000 designs. This service features an API with listing, search and registration services. Some of these services require an access token, for example, for searches with more than 100 results.

This API has some interesting features, such as: obtain information about a particular emoji; obtain a general index of emojis, which can be ordered by score, newest and popular, the last two being restricted to Pro/Premium users; and get the list of emoji categories. It also allows the user to add their own emojis. Pro and Premium memberships are some of the associated paid services.

The search for emojis is done using the emoji code. These searches can be carried out by codes that contain a term, start with a term, or end with a term. They can also be searched for by tags, or by the combination of a code search with tags to restrict the results to emojis that contain a term which also has the tags specified. The search can also be restricted to one or more categories.

The returned data for an emoji search is the following:

- `code`, the code of the emoji;
- `moji`, the character code of the emoji;
- `unicode`, the unicode ID of the emoji;
- `category`, the category the emoji is contained in;
- `tags[]`, an array of tags the emoji is registered under;
- `link`, URL associated with the emoji;
- `base`, the code of the emoji it is based on;
- `variants[]`, an array of variants/synonyms of the emoji;
- `score`, the score of the emoji;
- `r18`, specifies if the emoji contains adult/vulgar content;
- `customizations`, specifies overlays to customize the emoji;
- `combinations`, specifies combinations the emoji can be used in;
- `created_at`, creation/registration date in UTC format (Pro/Premium access only);
- `favorited`, amount of active favorites - for users who have the emoji marked as favorite (Pro/Premium access only).

In this API the search for emojis is quite intuitive, and it was possible to see information about the emojis searched for without problems.

[4] https://developer.emojidex.com/#api

4.2 Google-Images-Search

Google-Images-Search 1.3.3[5] is a library for image search that needs to have the Google Custom Search API enabled, needs API key credentials and a project set in place. After the initial setup, images can be searched and saved by defining search parameters. These parameters are:

- `q`, for the query to be searched;
- `num`, corresponding to the number of results to be displayed;
- `safe`, specifying the security level of the search (`high|medium|off`);
- `fileType`, to indicate the type of file (`jpg|gif|png`);
- `imgType`, allowing to detail the type of image (`clipart|face|lineart|news|photo`);
- `imgSize`, to define the size of the image (`imgSizeUndefined, HUGE, ICON, SMALL ...`);
- `imgDominantColor`, to designate the dominant color of the image (`black|blue, ...`);
- `rights`, allowing to specify image rights (`cc_publicdomain|cc_attribute| cc_sharealike|cc_noncommercial|cc_nonderived`).

In this case, the search is simple, but it presented results that are not always just icons, often including photographs. Even with the `clipart` parameter in the image type, the results are not suited to what is sought.

4.3 emoji-data

Project emoji-data[6] provides data about emojis with images for use on the web. It supports Emoji version 13 (2020) and has a catalog[7] with the emojis' data available. Their images are extracted from sources like Apple Emoji, Android Emoji, Twitter Emoji and Facebook Emoji.

The API data for the emojis was accessible, including their categories, but installation errors in the packages prevented the API from working properly.

4.4 Open Emoji API

Open Emoji API[8] is a free emoji API sourced directly from the Unicode public website[9]. With this API all emojis can be listed, as well as their categories, and the search for specific emojis is also available. For the search, the name of the emoji, or a term that is contained in the name, can be specified to get a single emoji or multiple corresponding emojis, respectively, and emojis

[5] https://pypi.org/project/Google-Images-Search/
[6] https://github.com/iamcal/emoji-data
[7] http://unicodey.com/emoji-data/table.htm
[8] https://emoji-api.com
[9] https://home.unicode.org

in a specific category or a list of all emoji categories can also be retrieved. Any of these searches returns the information of all the emojis that appear as a result. Information like:

- `slug`, the name of the emoji;
- `character`, the unicode that represents the emoji;
- `unicodeName`, the unicode name;
- `codePoint`, API code;
- `group`, the category of the emoji;
- `subGroup`, a subgroup of the category.

The emoji information was successfully accessed and the searched emojis were viewed.

4.5 Icons8 API

The Icons8[10] project creates tools and design elements to provide icons, illustrations, photos, music, and design tools. With an Api-Key header or a token parameter that specifies the user API to authenticate, the Icons8 API[11] allows the search to obtain their icons using the following search parameters:

- `term`, term to search;
- `amount`, how many objects to return;
- `offset`, how many objects to skip from the beginning of the returned results;
- `platform`, filter by platform;
- `language`, in which language to return the results;
- `token`, user API token;
- `authors`, possible values: `icons8` for icons from icons8 designers, `external` for icons from marketplace authors, and `all`;
- `isAnimated`, to include animated icons.

The returned results that include information about the icons are presented in a array of found icons for the specified parameters. They present properties such as: the unique `id` of the icon; the human-readable `name`; an API alias called `commonName`; the human-readable name of the `category` which contains the icon; the `platform` that also contains the icon; the indication of whether the icon is animated or not with `isAnimated`; and the indication of whether the icon `isFree` to use.

[10] https://icons8.com
[11] https://developers.icons8.com/docs/getting-started

4.6 IconFinder API

IconFinder[12] provides high-quality icons that can be accessed and integrated with the IconFinder API[13]. The search for icons can be done by a query string and some parameters to filter information. These parameters are:

- `query`, the word(s) to search;
- `count`, the number of icons to include in the result;
- `offset`, result offset of icons to show starting from 0, resulting in the first `count` icons being returned;
- `premium`, to filter premium icons;
- `vector`, to filter vector icons;
- `license`, to filter by license scope: `none` or `empty value` to include all icons regardless of the license; `commercial` to only include icons that can be used commercially; and `commercial-nonattribution` to include icons that can be used commercially without any attribution requirements;
- `category`, to filter icons based on a category;
- `style`, to filter icons based on an available style identifier;
- `is_explicit`, if false, it will exclude all explicit content. If true, it will only return explicit content.

In icon search parameters, excluding the search term, the options used are those that come by default from the APIs, which applies to all explored APIs, with the exception of `count`, which was set to 20, in order to restrict the number of results returned. The search returns a list of Icon objects and the number of icons returned in the result, with images of the icons available in Scalable Vector Graphics (SVG) and Portable Network Graphics (PNG) format.

4.7 Icons-50

Icons-50[14] is an image dataset of icons composed by 10,000 images from 50 classes of icons. Classes such as people, food, activities, places, objects, symbols, among others, gathered from different technology companies and platforms. This dataset contains data stored in a dictionary with the keys:

- `class`, representing 50 categories of icons with 10,000 elements in {0,1,...,49};
- `style`, defining the origin of the icon with 10,000 elements in {Apple, Google, etc.};
- `image`, with 10,000 3x32x32 images representing the icons;
- `rendition`, with 10,000 strings indicating the icon's version;

[12] https://www.iconfinder.com
[13] https://developer.iconfinder.com/reference/overview-1
[14] https://www.kaggle.com/danhendrycks/icons50

- subtype, with 10,000 elements specifying the subtype of a class such as whale or shark for the marine animals class.

The images were collected from emojipedia[15]. More details are presented in the research paper [53]. The emoji data is accessible, being possible to see the contents of the keys and search for emoji data in folders of certain categories.

4.8 ImageNet

ImageNet[16] is an image dataset which is organized according to the lexical database WordNet[17] hierarchy. The dataset explored was the Tiny Imagenet (Stanford CS231N)[18] for being the smallest among those available, therefore easier to work with. The dataset consists of three image folders (test, train and val) and two text files (words.txt and wnids.txt) with information about the dataset images. In the words.txt file of the dataset, we have access to an index that matches the folder name of the images in the train folder with their description. Therefore, it was the images in the train folder that were used to search for terms in the dataset.

Although it was possible to search for terms in the dataset, since this is a dataset with images and not icons, the results obtained from the search on ImageNet, presented next to icons, make the results very heterogeneous.

4.9 Unicode

According to the Unicode public website[19], Unicode is the universal character encoding, maintained by the Unicode Consortium. It provides the basis for processing, storage and interchange of text data in any language in all modern software and information technology protocols, having an unique number for every character, regardless of the platform, program, or language.

There are more than 136,000 Unicode characters, among which some are emojis. The list of the Unicode emoji characters and sequences for Emoji v13.1 can be seen in the online chart[20]. With new versions of Unicode Emoji[21], emoji characters and sequences are constantly being added. A detailed glossary of Unicode terminology can be found on the technical site[22]. This can also be a good resource to acquire icons, so it was further studied for the dictionary through the exploration of OpenEmoji API that uses the Unicode format.

[15] https://emojipedia.org
[16] http://image-net.org
[17] https://wordnet.princeton.edu
[18] http://image-net.org/download-images
[19] https://home.unicode.org
[20] https://unicode.org/emoji/charts/emoji-list.html
[21] https://www.unicode.org/emoji/charts/emoji-released.html
[22] https://unicode.org/main.html

4.10 Icon Search Evaluation

In an attempt to make a simple evaluation of an icon search with some of the icon sources mentioned above, to see the potentials and limitations using a story, we used Example (2):

(2) Three hooded individuals robbed a service station this Friday morning and ran away with money. One of the suspects wielded a shotgun to threaten employees and costumers in the space. The burglars fled on foot with the money. The police went to the scene.

Using spaCy and the Natural Language Toolkit (NLTK), we followed the steps presented below to search for icons:

1. Tokenization[23] with spaCy, by splitting the text into meaningful segments, called **tokens**. Here we take into account that it disregards the punctuation;

2. Lemmatization [57] and removal of stop words, punctuation and numbers. The aim of lemmatization is to reduce inflectional forms to a common base form, also known as **lemma**. In this case, the words in the document are lemmatized except if they are stop words, punctuation and numbers. In that case, they are removed.

   With spaCy we can access each word's base form with a token's `.lemma_` method and some of their attributes[24] with:

   `is_stop`: to check if the token is a **stop word**, that is, if they are the most common words in a language like "the", "a", "on", "is", "all". These words do not carry important meaning so they can be removed from texts;

   `is_punct`: to check if the token is punctuation to avoid returning tokens that are punctuation;

   `like_num`: to check if the token represents a number.

3. Find the icons that correspond to the words in the text. If the icon corresponding to the word being searched for is found immediately, then that icon is returned. Otherwise, we search for icons corresponding to a list of synonyms of the initial word. This list of synonyms is derived using WordNet[25] from NLTK.

This evaluation using Example (2) was made for some of the previously presented icon sources. The icon search results are shown in Figure 4.1a regarding emojidex, Figure 4.1b on Open Emoji API, Figure 4.1c about Icons8 API, Figure 4.1d on IconFinder API, Figure 4.1e for Icons-50 dataset, and a summary of all results is presented in Table 4.1.

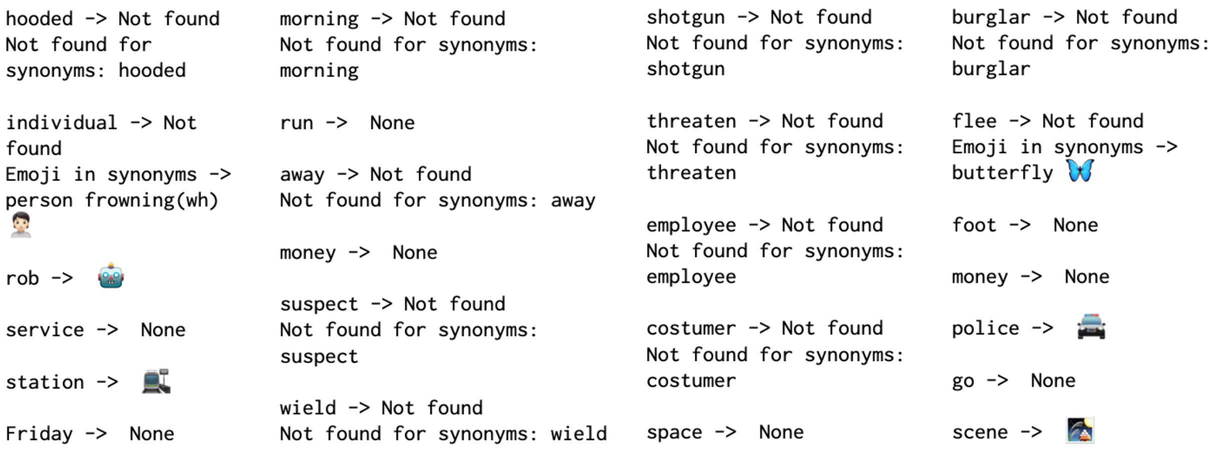
Results from the Google-Images-Search library and the ImageNet dataset were not included in this evaluation due to the fact that the resulting images in both sources are very heterogeneous.

[23]https://spacy.io/usage/linguistic-features#tokenization
[24]https://spacy.io/api/token#attributes
[25]https://www.nltk.org/howto/wordnet.html

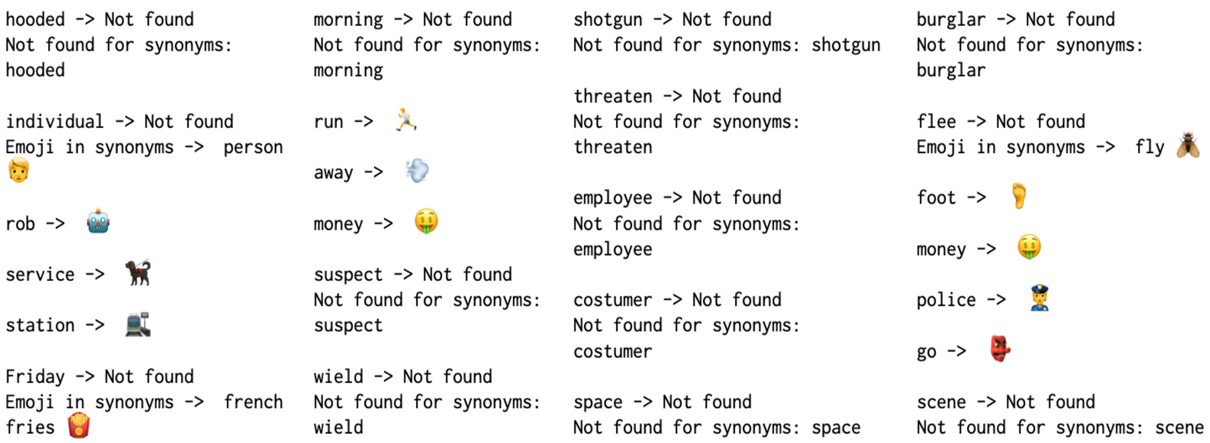
API	Icons expected	Icons found (total)	Icons correctly found	Icons correctly found from synonyms list	Icons incorrectly found from synonyms list
emojidex	24	6	2	1	1
Open Emoji API	24	13	6	1	2
Icons8	24	23	16	0	1
IconFinder	24	24	15	0	0
Icons-50	24	4	3	0	1

Table 4.1: Search results for icons.

In Google-Images-Search, it is difficult to restrict the search to icons, and the ImageNet dataset has a mix of photographs with images, making it difficult to filter the results.



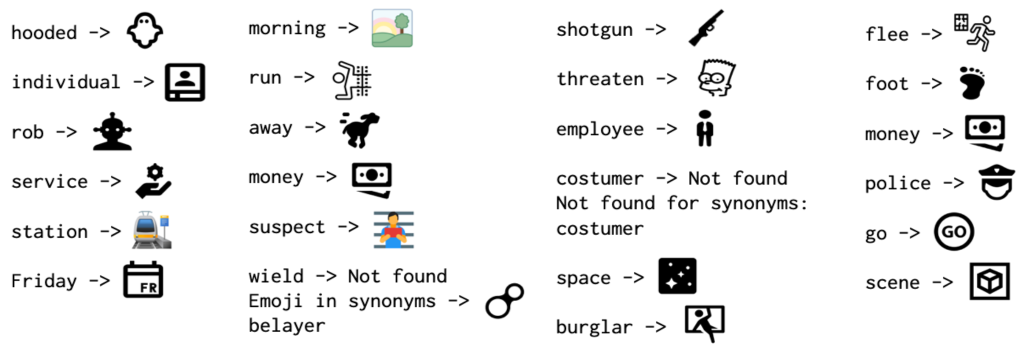
(a) emojidex.



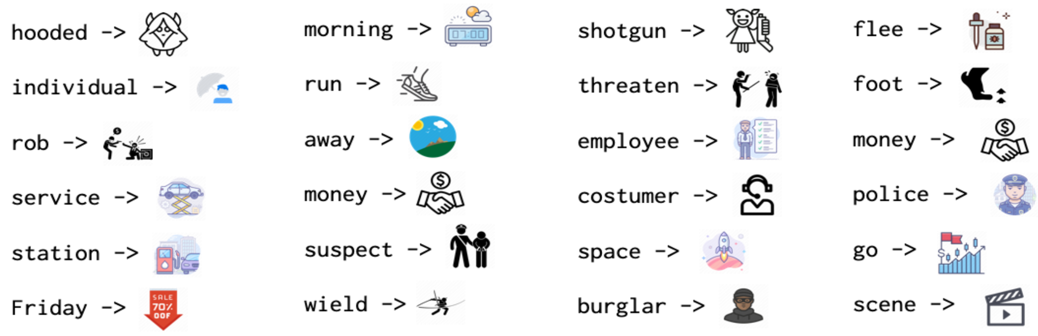
(b) Open Emoji API.

Figure 4.1: Icons obtained by the evaluation using icon sources.

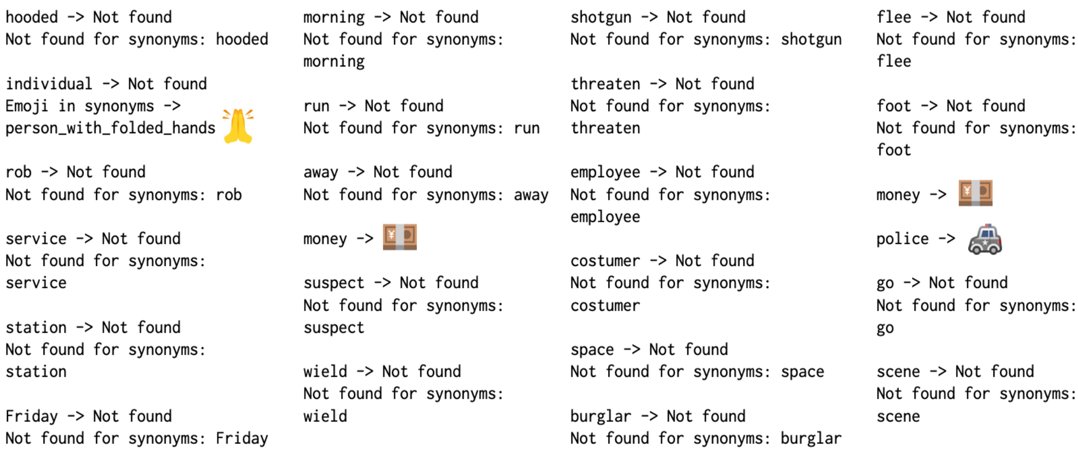
Even with simple text processing, such as the one used in this evaluation, this type of assessment helps to realize some limitations when searching for icons using the researched sources.



(c) Icons8 API.



(d) IconFinder API.



(e) Icons-50.

Figure 4.1: Icons obtained by the evaluation using icon sources.

We also present Table 4.2 with an overview of the important information about the icon sources covered in this chapter, in order to have a generalized view of them all by summarizing them.

Source	Type of Source	Authentication	Usability Limits (for free)	Results Format	Items Format	Usage License	Notes
emojidex	API	User auth token	No clear information about it.	JSON	SVG or PNG (regular and animated, with PNG in various sizes)	Open, Closed, and Commercial Print and Tangible Goods License[a].	Newest and popular emoji search available for Pro/Premium users.
Google-Images Search	API	API Key	10 images per request.	BytesIO object	JPEG, GIF and PNG	MIT License (MIT)	Results may include photographs.
Open Emoji	API	API Key	No clear information about it.	JSON	Unicode	N/A	-
Icons8	API	API Key or token	50 Monthly downloads; 250 Other calls.	JSON	SVG or PNG	Non-transferable, non-exclusive worldwide license to use the icons, photos, and illustrations for the Permitted Uses[b].	3 types of paid options for 1,000, 10,000 and 100,000 monthly downloads.
IconFinder	API	API Key	Access to 322,025 free icons, with a rate limit depending on the authentication method.	JSON	SVG or PNG in various sizes	Limited, non-exclusive, non-transferable, non-sublicensable, revocable right and license during the Term of Agreement and under the intellectual property rights, to access and use the Services[c].	5,000,000 free and premium icons available with payment plans; Provides an online icon editor.
Icons-50	Dataset	Not required	Access to the complete dataset.	N/A	PNG	N/A	Dataset available to download.
ImageNet	Dataset	Profile with Accesskey	Access to the complete datasets.	N/A	JPEG	Non-commercial research and educational purposes[d].	Dataset available to download.

Table 4.2: Summary of features on the covered icon sources.

[a] https://www.emojidex.com/emojidex/emojidex_general_license_terms_and_conditions
[b] https://developers.icons8.com/license
[c] https://support.iconfinder.com/en/articles/18224-terms-of-service-for-api
[d] https://image-net.org/download.php

By analyzing the tables and images above, we verified that, in terms of restrictions on use and access to icons, the datasets and APIs are quite different. With datasets, there is early access to all available content, whereas using free versions of the APIs, access can be revoked depending on the number of API accesses already made or the number of icons downloaded.

It is also clear that good icon search results will depend on the quantity and quality of icons available in the sources used for the search. For this reason, the integration of more than one source can lead to better results, increasing the range of options, and enabling the comparison of results obtained, thus making it possible to choose the one that best suits the search.

4.11 Summary

An overview of the characteristics of the sources researched is presented, as well as their functionalities and potential use for the desired purpose: the construction of an icon dictionary for use in the final visualization.

To summarize, according to the research presented in this chapter, we can have three possible types of icons that the dictionary may support depending on its source:

1. **url**, if it comes from an icon API, for example;
2. **local**, if it is an icon from a local directory, for instance, from a local dataset;
3. **unicode**, if it is an icon accessed through its unicode.

Any of these icon sources options has its challenges, which will be discussed further on when presenting the work developed in the construction of the dictionary.

Chapter 5

Pipeline Description

Our demonstration pipeline aims to transform text into icons, with our contribution being focused on the visualization step by adding a new visualization using icons.

As stated in Chapter 1, our visualization is integrated in the Brat2Viz project pipeline, making it essential to understand which steps compose it. These steps consist of the annotation, and the formal and structured representation of the narrative elements, overviewed in Section 5.1. In Section 5.2, the modules into which the pipeline is divided are explained. And finally, in Section 5.3, the visualizations already supported by the project are presented.

Figure 5.1 will be useful throughout the chapter to understand where the covered topics fit into the pipeline, as it represents the steps, from the annotation of the text to the visualization.

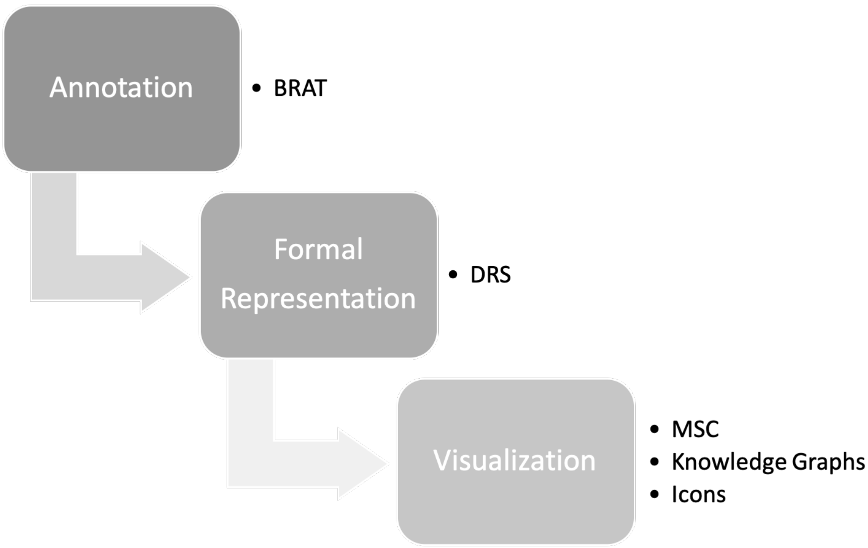


Figure 5.1: Pipeline steps.

5.1 Annotation and Formal Representation: Overview

In the process of extracting narratives from texts, it is necessary to identify: narrative elements, such as actors, events and times; their attributes; and semantic links between them, such as temporal, referential, and semantic roles. This results in a structure that can then be represented using a formal logic-based language. From this structure, different representations of the narrative can be produced, such as diagrams or text.

Brat2Viz[1] is a Narrative Annotation Visualization tool [4] proposed to support the debugging of narrative annotation done with the Brat Rapid Annotation Tool (BRAT), thus reducing the human effort required for the task, and supporting the high level visualization of narrative annotation. Brat2Viz implements a pipeline from annotated text to visualization, by transforming the annotation into a formal representation, using Discourse Representation Structure (DRS), and then, to visual representations. At this time, the supported visualizations are: Message Sequence Charts (MSCs) and Knowledge Graphs, which can be seen in the interactive online demonstration of the pipeline.

This section aims to give an overview of the work previously done in the pipeline, by explaining its features and providing insight on the annotation and formal representation used for the visual representations in Brat2Viz, as they are fundamental to the integration of a new visualization.

5.1.1 Annotation

The annotation of the project *corpus* has three levels: referential, temporal and semantic role labeling. The project annotation scheme [107] consists of the harmonization of three different standards: ISO standards (24617-1/9) [42, 43] used in the first two levels, and Linguistic InfRastructure for Interoperable ResourCes and Systems (LIRICS) [4] used in the third level.

The *corpus* annotation task includes choosing a semantic annotation framework, in this case BRAT; adapting to properties of the target language, which for this project is European Portuguese; and some adaptations related to the number of tags and types of attributes to use, due to the genre of the *corpus* (news stories), and to the multilayer annotation. The multilayer semantic annotation framework is divided in: entity structures (events, times, and participants), and link structures (temporal, aspectual, subordination, objectal, and semantic role links).

All of this results in the current annotation scheme with three types of general annotation categories (**tags**) with attributes (**sub-tags**) that give a more complete meaning to the annotated components. These are:

- *Actors* - for characters (e.g., “um homem” - “a man”);
- *Events* - for events (e.g., “fugiu” - “ran away”);

[1] https://nabu.dcc.fc.up.pt/brat2viz

- *Timex3* - for temporal expressions (e.g., “terça-feira à tarde” - “Tuesday afternoon”).

To guarantee a proper connection between the actors, the events and the actors, and between the events, temporal references and other objects or locations, three types of links are used:

- *TLINKS* - Temporal Links that take into account the chronological ordering of the events, which allows us to observe if one event happens before, after or at the same time as another. It is also used to represent temporal relations between events and temporal expressions;
- *REF_REL* - Referential Relations that denote the relations between actors by representing the lexical relations between linguistic units, such as synonymy, antonymy, hyponymy, etc;
- *OBJ_REL* - Objectal Relations also denote the relations between actors, but represent relations between linguistic units, from a discourse point of view.

In order to establish thematic relations, a list of essential semantic roles was also adapted from the concepts of LIRICS [86].

As previously mentioned, BRAT [112] is the annotation framework used in the project. This web-based tool supported by Natural Language Processing (NLP) technology for text annotation, allows an annotator to add notes to existing text documents, and is available in an online environment[2] for collaborative text annotation.

Some of the categories of annotation supported by BRAT are: text span annotations, such as `Organization` and `Person`, which are suitable to create annotations for Named Entity Recognition (NER); relation annotations for simple relational Information Extraction (IE) tasks, like the `Family` relation that connects several `Persons`; and the annotation of n-ary associations by linking several annotations that can be used, for example, for event annotation. The use of attributes on the annotations, like marking an event as factual or speculative, specify types and properties of the annotations.

At the end of the annotation process, the annotated document data is exported and viewed in the annotated files generated by BRAT (`.ann`). Each news story has its own annotated file.

5.1.2 Formal Representation

Discourse Representation Theory (DRT) [45] is a theoretical framework with theoretical principles to ascertain meaning in natural languages that introduced representations, called DRSs. This is the format in which the encoded information is represented. DRS is a declarative, logic-based, intermediate language used as a representation scheme that can be shaped according to the intended purpose.

[2] https://brat.nlplab.org

The use of this formal logic-based language serves as a basis for the construction of the visualization of news stories, with a DRS file being created for each news. These DRS files are composed of statements generated to each expression outlining in a textual format, the properties of the events, the actors, the temporal expressions, and the relationships between them. With this representation, it is not necessary to return to the original text in the annotated file, as it is possible to access the information through operations to visualize, evaluate, and rewrite the necessary information, which allows us to infer information about the narratives.

5.2 Brat2Viz Modules

Brat2Viz[3] is composed of two modules: Brat2DRS and DRS2Viz. In order to exemplify the details given below about the steps of the modules, from the news story to the visualization, consider Example (3) as an illustrative news story:

(3) Um homem assaltou um casal que estava dentro de um carro, apontando-lhes uma pistola, na avenida da Liberdade, Lisboa. O crime ocorreu terça-feira à tarde e o ladrão fugiu com cerca de 400 euros. As vítimas encontraram o assaltante no Rossio. O ladrão terá conseguido convencer ambos de que os conhecia, entregando-lhes uma câmara de filmar e dois relógios. O casal foi então convidado a ir para o carro, onde foi assaltado.

The two modules of Brat2Viz:

- **Brat2DRS**: following the annotation scheme, the module creates a DRS representation for each news story, by parsing the annotation files generated by BRAT (`.ann` files), and building a dictionary with the linguistic elements of the text where it associates a symbolic variable to each event. Thus converting the interpreted annotations to a DRS using the NLTK library[4].

  The generated DRS statements are produced for each expression containing the events and their properties, actors, temporal expressions and their relations. In Figure 5.2a we can see a sample of the annotation of the first sentence of Example (3) and in Figure 5.2b the respective generated DRS output is presented.

  The narrative represented in this way means that new operations do not need to go to the annotated file, making it unnecessary to go back in the process.

- **DRS2Viz**: consists of a parser and a visualization engine. This module receives the DRS representations of Brat2DRS to deploy a web application with the generated visualizations of the news text.

  The parser uses the DRS to extract actors, events and relationships into independent data structures, while keeping track of their identifiers. Since the same actor can appear

[3] https://github.com/LIAAD/brat2viz
[4] https://www.nltk.org/howto/drt.html

```
T1 ACTOR 0 8 Um homem
A1 Lexical_Head T1 Noun
A2 Individuation T1 Individual
A3 Actor_Type T1 Per
A4 Involvement T1 1
T2 EVENT 9 17 assaltou
E1 EVENT:T2
A5 Class E1 Occurrence
A7 Tense E1 Past1
A8 Polarity E1 Pos
A9 Time E1 Past
T3 ACTOR 18 26 um casal
A10 Lexical_Head T3 Noun
A11 Individuation T3 Set
A12 Actor_Type T3 Per
```

(a) Annotation.

```
# E1 (assaltou) -> a
# FOL: exists Factual Occurrence Past Past1 Pos a.(event(a) &
eventClass(a,Occurrence) & Tense(a,Past1) & Polarity(a,Pos) &
Time(a,Past) & occursBefore(a,f) & relationRole(agent,T1) &
relationRole(theme,T3) & Factuality(a,Factual))
# DRS: ([a],[event(a), eventClass(a,Occurrence),
Tense(a,Past1), Polarity(a,Pos), Time(a,Past), occursBefore(a,f),
relationRole(agent,T1), relationRole(theme,T3),
Factuality(a,Factual)])
```

(b) DRS.

Figure 5.2: First lines of the annotation of Example (3) and a snippet of the respective DRS output.

described in different ways throughout the news story, it is necessary to deal with redundant actors. To do this, the parser merges them into single actors, updating their references in events and relations structures. After the parser is completed, the visualization engine generates the visualizations supported by the project, discussed below.

5.3 Supported Visualizations

As stated before, the currently implemented visualizations are MSC and Knowledge Graphs. Both represent the actors as nodes, whereas events and relationships are represented as links between the nodes.

5.3.1 Message Sequence Chart (MSC)

MSCs compose an intuitive visual representation that is widely used in the field of software engineering. More recently, the automatic construction of MSCs using NLP has been addressed. In software engineering they are used to capture system requirements during the early design stages in domains such as telecommunication software [51]. In NLP, it allows the representation of actors and interactions among them in a scenario [54].

There are some works with similarities to the work proposed in this project. As an example, the work presented by Palshikar et al. [81] adopts MSC to represent annotations. They applied non-supervised approaches using dependency parsing and Semantic Role Labeling (SRL) to history narrative extraction, and then used MSC to illustrate the obtained results. Another work proposed by Hingmire et al. [54], uses MSCs as a representation to visualize narrative text in Hindi, by presenting an approach to extract MSC actors and interactions from a Hindi narrative.

The work presented in Brat2Viz, uses manually annotated data, does not consider simplifications regarding narrative elements, applies the technique in a news story *corpus*, and also proposes the use of DRS as an intermediate step between a text in any language and a visual language, in order to prevent possible linguistic ambiguities.

This visualization is generated using mscgen_js[5], a Javascript implementation of MscGen[6] that renders message sequence charts from MSC strings. The complete MSC output generated for Example (3) can be seen in Figure 5.3, being presented in Figure 5.4 a part taken from that same output, where the elements of the visualization can be seen in more detail.

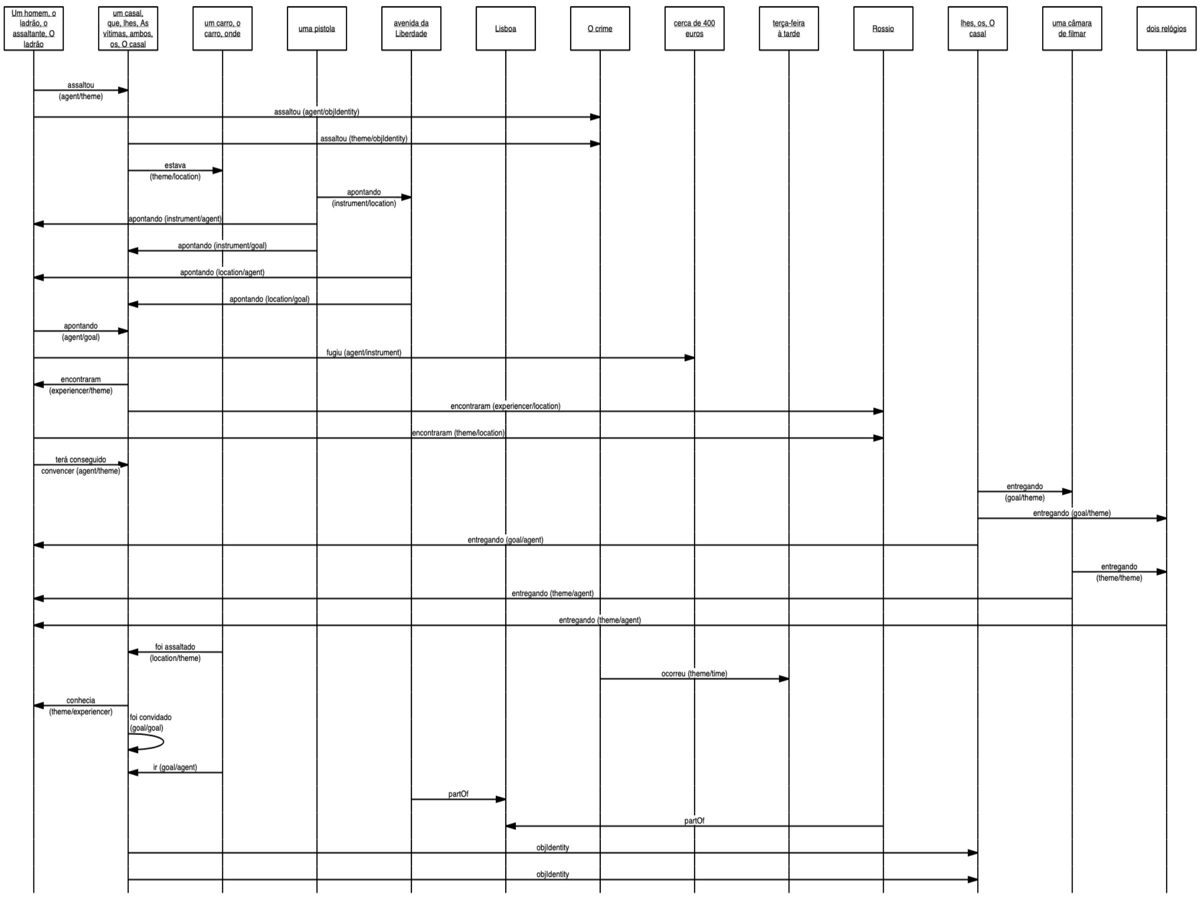


Figure 5.3: MSC representation of Example (3).

[5] https://github.com/sverweij/mscgen_js
[6] http://www.mcternan.me.uk/mscgen/

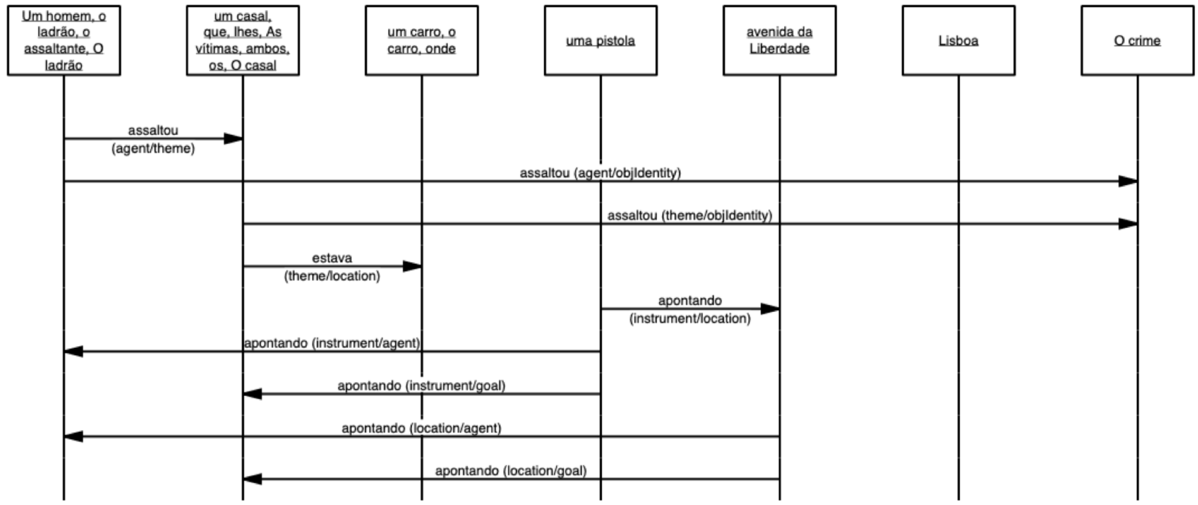


Figure 5.4: Part of the MSC representation of Example (3).

5.3.2 Knowledge Graphs

As presented by Ehrlinger et al. [34], there are several definitions and possible interpretations available on knowledge graphs, from the mention of Google's Knowledge Graph [108], to several definitions published in articles [44, 83, 91].

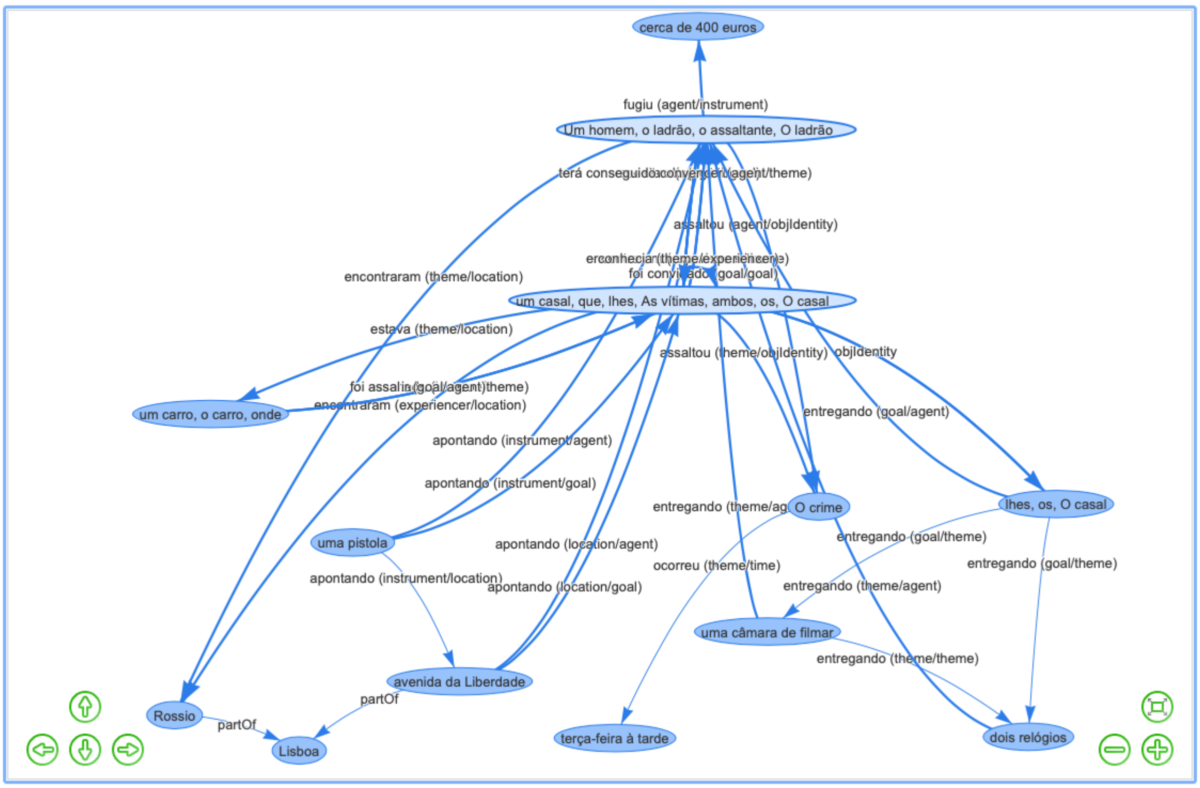


Figure 5.5: Knowledge Graph representation of Example (3).

In this sense, Ehrlinger et al. [34] propose a definition of knowledge graph taking into account several definitions and applications, leading to a definition that distinguishes it from similar terms. The definition presents a knowledge graph as something that acquires and integrates information into an ontology, applying a reasoner to generate knowledge.

Within the scope of Brat2Viz, the knowledge graph serves to store data resulting from an information extraction task that can be defined as a set of nodes and edges. This visualization is generated using visjs[7], a Javascript library used to build, customize, and display networks. Figure 5.5 shows the graph generated from Example (3), where we can see the buttons on the left and right of the image, which allow the user to interact with the graph, moving it and zooming in and out. It is also possible to move both the nodes and the edges freely.

5.4 Summary

Now with a more in-depth overview of the project, it becomes clearer where the visualization using icons fits in the pipeline. The DRS format file with the information regarding the news story will be used as input for the visualization, in order to have this information adequately portrayed, by presenting as output a set of icons representing the news story. Therefore, it is important to explain the focus of our contribution to the pipeline by describing the necessary steps taken to achieve this new visualization.

[7] https://github.com/visjs/vis-network

Chapter 6

Extraction of Narrative Elements

As we have come to understand, in order to represent a story, regardless of the desired visualization, it is necessary to identify and extract the required information to make an adequate representation.

This chapter describes what needs to be done to achieve the icon visualization, regarding the extraction of narrative elements, the necessary processing for these same elements, as well as the implementations[1] made. We start by explaining the need for sentence extraction in Section 6.1, followed by the translation process in Section 6.2. Lastly, in Section 6.3, we present the algorithm used to obtain the more specific description of the narrative's actors. In this section, we also apply the algorithm to an example and evaluate its correctness. The steps of the process described in this chapter are exemplified in Figure 6.1.

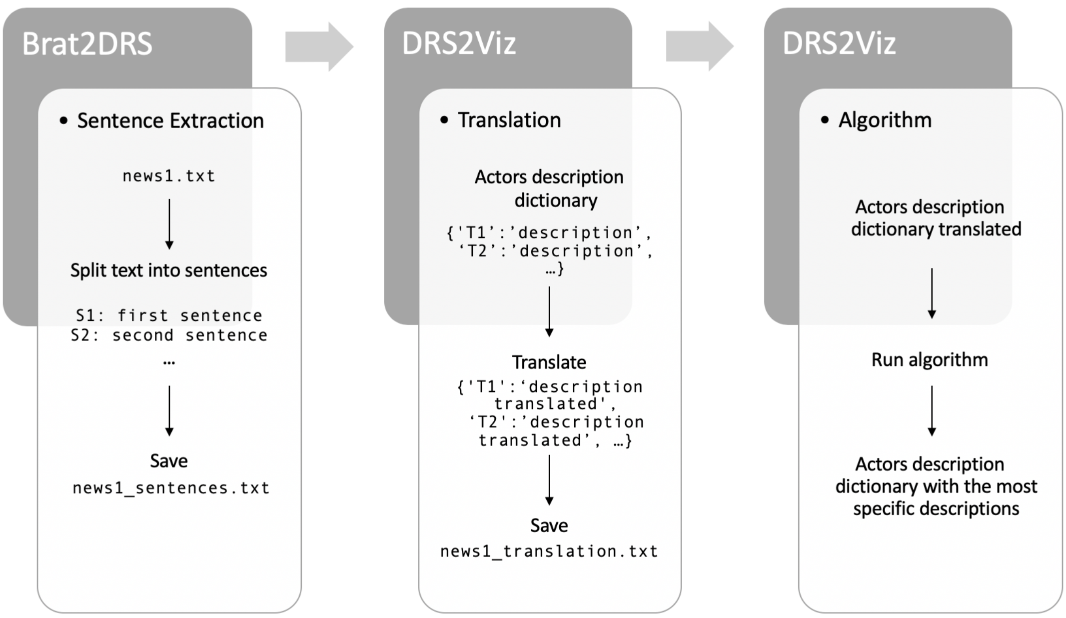


Figure 6.1: Extraction steps.

[1]Available at: https://github.com/LIAAD/Text2Icons

6.1 Sentence Extraction

Similar to the representation used in the Graphical Storytelling[2] project in which news comics are generated for journalistic texts, the idea for the icon visualization was to have a graphic representation for each sentence in the news story.

At the start of the work presented in this thesis, a limitation was found in the information contained in the Discourse Representation Structure (DRS) files. Since there was no information about which actors, events, temporal expressions and other relations belonged to each sentence, and the information was related to the entirety of the news story, it became necessary to extract the sentences of each news story at the beginning of the pipeline.

An implementation has been added to the Brat2DRS module that automatically extracts all the sentences that make up the project's news collection. Following the same format used for the annotation of the linguistic elements of the text, identifiers were assigned to each sentence ($S1, S2, \ldots$), saving each identifier with the respective sentence in new files with the sentences of each news story.

6.2 Translation

Since the researched sources for the icon search have their information in English and the project works with news stories in Portuguese, it was necessary to think of a form to automatically translate the stories and integrate this with the process. Two translation methods have been implemented in the DRS2Viz module, which are available to use in the icon visualization, and are presented and explained in the following subsections.

6.2.1 Translation Methods

We now present the two translation methods that have been implemented in the DRS2Viz module.

- **Automatic translation using Hugging Face Transformers[3]:**

  Transformers[4] is an open-source library that includes tools to facilitate training and development, allowing users to access large-scale pre-trained models that are easy to read, extend, and deploy. These models are capable of performing tasks on texts such as classification, Information Extraction (IE), question answering, summarization, translation, among others, in over 100 languages.

[2] https://bbcnewslabs.co.uk/projects/graphical-storytelling/
[3] https://github.com/huggingface/transformers
[4] https://huggingface.co/transformers/

It is designed to reflect the standard Natural Language Processing (NLP) machine learning model pipeline that processes data, applies a model, and makes predictions. Each model is composed of: a **tokenizer**, to convert raw text to sparse index encodings; a **transformer**, which transforms sparse indices to contextual embeddings; and a **head** that uses the contextual embeddings to make a task-specific prediction [129].

As the task to be performed is translation, the chosen model was `Helsinki-NLP/opus-mt--roa-en`[5] that translates Romance languages to English, this way we can translate the actors' descriptions from Portuguese to English.

To download and use the pre-trained model for our task we used PyTorch[6]. The code used is presented in code block 6.1.

```
from transformers import AutoTokenizer, AutoModelForSeq2SeqLM

tokenizer = AutoTokenizer.from_pretrained("Helsinki-NLP/opus-mt-roa-en")
model = AutoModelForSeq2SeqLM.from_pretrained("Helsinki-NLP/opus-mt-roa-en")
```

Listing 6.1: Load model.

The `AutoTokenizer`[7] is a generic tokenizer class that was instantiated as one of the tokenizer classes of the library. In turn, the `AutoModelForSeq2SeqLM`[8] is a generic model class that was instantiated as one of the model classes of the library, with a sequence-to-sequence language modeling head.

Afterwards, the text was run through the model and the resulting embeddings were decoded. The final result is the text translated to English. The code necessary for this is in code block 6.2.

```
res = model.generate(**tokenizer(txt, return_tensors="pt", padding=True))
translation = [tokenizer.decode(t, skip_special_tokens=True) for t in res]
```

Listing 6.2: Translate text with Hugging Face Transformers.

- **Automatic translation using Googletrans**[9]:

  Googletrans is a free Python library that implements Google Translate Application Programming Interface (API), using the Google Translate Ajax API to make calls to some methods. Some of the interesting features of this library are the fact that it uses the same servers as Google Translate website[10], has automatic language detection, and does bulk translations.

[5] https://huggingface.co/Helsinki-NLP/opus-mt-en-roa
[6] https://pytorch.org/get-started/locally/#start-locally
[7] https://huggingface.co/transformers/model_doc/auto.html#autotokenizer
[8] https://huggingface.co/transformers/model_doc/auto.html#automodelforseq2seqlm
[9] https://pypi.org/project/googletrans/
[10] https://translate.google.com

The simple implementation that allows the translation of a text using Googletrans, can be seen in code block 6.3.

```
from googletrans import Translator

translator = Translator()
translation = translator.translate(txt, src='pt', dest='en')
translated_text = translation.text
```

Listing 6.3: Translate text with Googletrans.

`Translator` is the Google Translate AJAX API implementation class. It is necessary to create an instance of `Translator` to use this API. The function `translate` is used to translate text from a source language to a destination language. If a source language is not provided in the parameters, Google Translate attempts to detect the source language, just like the web application does.

6.2.2 Saved Translations

When running the visualization for a selected news story, it is necessary to translate the actors' descriptions. In order to make this process more time efficient, when a news story is selected, its translation is searched in a specific directory with a folder for the saved translations. If a translation file already exists for the selected news story, it means that its translation has already been done, and it is only necessary to fetch the translation from the respective file. If not, one of the available translation methods is used to carry out the translation, and then a new file with the translation is saved in the translations folder. In terms of space it is less efficient, but in general we can afford the trade-off.

6.3 Actor Generality Level Resolution Algorithm

One of the challenges that arose when trying to visualize a narrative using icons was the fact that the same actor can be described in different ways throughout the narrative. It then became important to identify the most specific description that defines an actor, in order to search for that description, thus trying to ensure that we search for the icon that best represents the actor.

For this, we propose a search algorithm that, from a set of descriptions of an actor in the narrative, finds the most specific description that describes it, using WordNet[11] similarity between words and the super-subordinate relation, also called hyperonymy and hyponymy.

According to the online Cambridge Dictionary[12], a **hyponym** is a word whose meaning is included in the meaning of another word, and a **hypernym** is a word whose meaning includes

[11] https://wordnet.princeton.edu
[12] https://dictionary.cambridge.org

a group of other words. For example, "horse" is a hyponym of "animal" and "animal" is a hypernym of "dog".

6.3.1 Algorithm

The algorithm receives a dictionary extracted from the corresponding news story DRS file, with the actor identifier as the key and the set of actor descriptions as the value. The algorithm works with the value, that is, the set of actor descriptions, returning at the end of the process a dictionary with the actor's identifier and its most specific description. This implementation was added in the DRS2Viz module.

The steps taken to obtain the most specific description, from the initial example with a set of actor descriptions to the most specific are presented below, where the methods mentioned belong to the WordNet Interface[13].

1. Convert all uppercase characters to lowercase;
2. Lemmatization process, removal of stop words, punctuation and numbers, as described in 4.10. This is achieved using spaCy's English pipeline `en_core_web_lg`[14];
3. Eliminate duplicate elements from the resulting list of step 2;
4. Filter the list obtained in step 3 to only have elements that have Noun Synsets. A **synset** is a set of synonyms that share a common meaning, where each synset contains one or more lemmas, which represent a specific sense of a specific word. This way we ensure that we have a list with words that exist in WordNet;
5. Filter the list obtained in step 4 to only have elements that have Noun Lemmas (synonyms within each sense);
6. Go through the list of terms obtained from step 5 and compare the elements two by two to get a list of hypernyms, using the method `lowest_common_hypernyms()` to get the lowest single hypernym that is shared by two words;
7. Compare the elements of the list of hypernyms from step 6 with the `root` node to get the most specific hypernym. For that, the similarity is calculated i.e., the distance to the `root`, using the method `path_similarity()` that returns a score denoting how similar two words are. The score ranges from 0 to 1, with 1 representing the identity i.e., comparing a word with itself. Therefore, the farther away from the `root`, the more specific it is.

   We consider actors to be nouns, and the `root` node for nouns is called `entity`.

   When the similarity score is the same, it means that they are at the same distance from the root node. We then proceed to the calculation of the amount of hyponyms that each one

[13] https://www.nltk.org/howto/wordnet.html
[14] https://spacy.io/models/en#en_core_web_lg

has, to break the tie. We consider the one with the least amount of hyponyms to be the most specific because it has fewer descriptions for itself. This is done through the method `hyponyms();`

8. Calculate the similarity between the elements of the list after step 5, and the lowest common hypernym, calculated in step 7. This is so we can obtain the element with the highest score of similarity to the lowest common hypernym, that is, the most specific element.

6.3.2 Example

For the purpose of exemplifying the steps of the algorithm, consider the following Example (4) as a set of descriptions of actor T1:

(4) `{'T1':'A man, the thief, the assailant, The thief, The guy'}`

Applying the steps of the algorithm described above to the set of descriptions of Example (4), we have the following step-by-step results:

Step 1: `a man, the thief, the assailant, the thief, the guy`

Step 2: `['man','thief','assailant','thief','guy']`

Step 3: `['man','thief','assailant','guy']`

Step 4: `['man','thief','assailant','guy']`

Step 5: `['man','thief','assailant','guy']`

Step 6:

```
      hypernym(man,thief) : person
  hypernym(man,assailant) : person
        hypernym(man,guy) : man
hypernym(thief,assailant) : wrongdoer
      hypernym(thief,guy) : person
  hypernym(assailant,guy) : person
```

List of hypernyms: `['person','person','man','wrongdoer','person','person']`

List of hypernyms (with no duplicates): `['person','man','wrongdoer']`

Step 7:

```
   similarity(person,entity) : 0.25
      similarity(man,entity) : 0.16666666666666666
similarity(wrongdoer,entity) : 0.16666666666666666
```

Most specif hypernyms: `['man', 'wrongdoer']`

Number of hyponyms of `man`: 51

Number of hyponyms of `wrongdoer`: 22

Most specif hypernym (after the tiebreaker): `['wrongdoer']`

Step 8:

```
        similarity(man,wrongdoer) : 0.2
      similarity(thief,wrongdoer) : 0.25
  similarity(assailant,wrongdoer) : 0.5
        similarity(guy,wrongdoer) : 0.16666666666666666
```

Final list with the most specific description: `['assailant']`

6.3.3 Results

To evaluate the effectiveness of the algorithm described above and to evaluate which translation to use, 8 examples of annotated news articles with 109 sets of actor descriptions were analyzed.

The data presented in Table 6.1 comes from a manual analysis of the results obtained from processing the actors' descriptions using the algorithm and comparing them with the expected descriptions. To carry out this evaluation, two datasets were created: one for the translations obtained using Hugging Face Transformers and the other for those obtained using Googletrans. These datasets facilitate the evaluation of the algorithm, allowing to run several examples of news stories at the same time. The implementation[15] for the evaluation counts for each news story the number of:

- Actors present in each news story;
- Well defined actors by the algorithm;
- Wrongly defined actors by the algorithm;
- Undefined actors by the algorithm;
- Total number of all previous counts.

6.3.4 Discussion of Results and Limitations of the Algorithm

When we analyze the results shown in Table 6.1 and summarized in the graph in Figure 6.2, we notice that the examples that were translated by Googletrans are the most favorable. While

[15] Available at: https://github.com/LIAAD/Text2Icons

ID	Actors	Well defined Actors				Wrongly defined Actors				Undefined Actors			
		Hugging Face		Google Translate		Hugging Face		Google Translate		Hugging Face		Google Translate	
1	13	11	10.09%	11	10.09%	2	1.83%	1	0.92%	0	0.00%	1	0.92%
2	14	10	9.17%	11	10.09%	4	3.67%	3	2.75%	0	0.00%	0	0.00%
3	13	6	5.50%	9	8.26%	5	4.59%	2	1.83%	2	1.83%	2	1.83%
4	20	12	11.01%	11	10.09%	7	6.42%	8	7.34%	1	0.92%	1	0.92%
5	10	5	4.59%	7	6.42%	5	4.59%	3	2.75%	0	0.00%	0	0.00%
6	15	10	9.17%	11	10.09%	5	4.59%	4	3.67%	0	0.00%	0	0.00%
7	6	3	2.75%	3	2.75%	3	2.75%	3	2.75%	0	0.00%	0	0.00%
8	18	10	9.17%	11	10.09%	6	5.50%	6	5.50%	2	1.83%	1	0.92%
Total	109	67	**61.47%**	74	**67.89%**	37	**33.94%**	30	**27.52%**	5	**4.59%**	5	**4.59%**

Table 6.1: Algorithm application to automatically translated examples with Hugging Face Transformers and Googletrans.

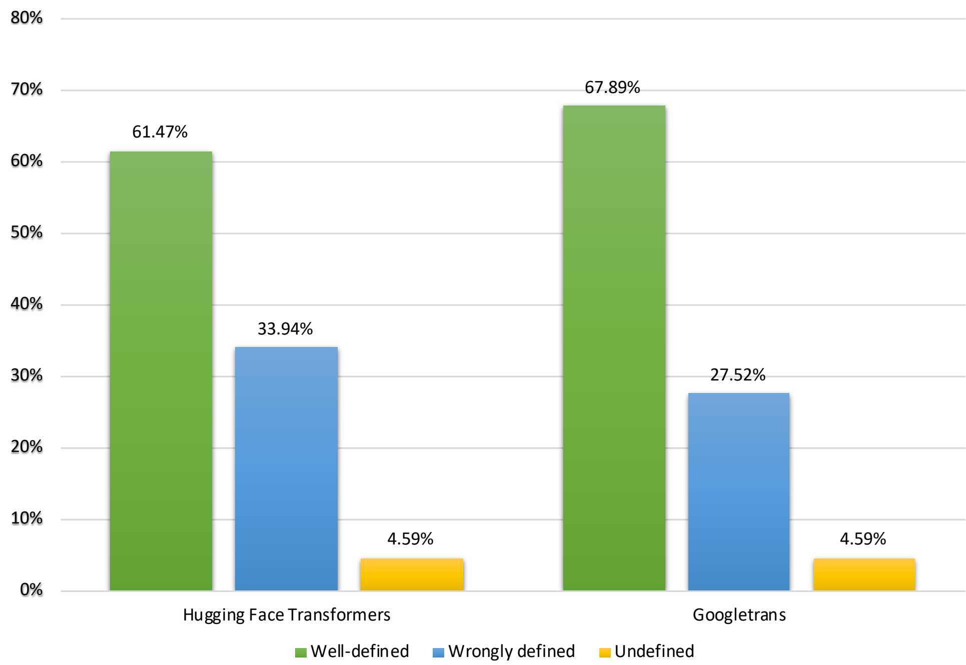


Figure 6.2: Actor resolution algorithm evaluation results.

running the code for both translations we also realized that the translation with Hugging Face Transformers is slower than using Googletrans.

One of the problems that we encountered through the analysis of some of the actors that emerged wrongly defined or undefined, was due to the fact that they are not being identified as people or places, to name a few reasons. More specific actors tend to not be recognized by WordNet and later on to not have an adequate match when searching the icon sources for an icon that represents them.

Another limitation of this algorithm is the fact that it restricts the description of actors to just one term. Actors that are composed of more than one term and cannot be reduced to just one without losing their meaning, end up getting lost in this process. An example where this happens is the actor "Avenida da Liberdade", which after being translated and processed by the algorithm results in "liberty". As we can see, we lost the original meaning of the actor and when searching for a matching icon in the icon dictionary, we get the Statue of Liberty as a result. Thus getting an unreliable representation of that specific actor .

6.4 Summary

In this chapter, the necessary narrative elements for the new visualization using icons were presented: the extraction of sentences from the stories, translation methods, and the algorithm that chooses the most specific description of the actors.

In the next chapter we will see how the visualization was constructed, presenting how these narrative elements were incorporated and describing the necessary implementation.

Chapter 7

Icons From Narratives

In the previous chapter, we obtained the necessary information to proceed with the visualization, through the extraction of narrative elements. Therefore, the steps taken to reach the icon visualization are described in this chapter, by explaining the implementation[1] made throughout this thesis.

We start with an overview of how the implementation is organized in Section 7.1. In Section 7.2 the icon dictionary developed is presented, explaining its construction, information and features. Next, in Section 7.3, we clarify how the icon dictionary was integrated in the visualization step and explain the development of the visualization. Finally, in Section 7.4, the results obtained are evaluated, analyzing the ability of linking terms to icons and the quality of the story representation with icons.

7.1 Implementation Overview

The backend part of the implementation, which makes up the necessary data processing to generate the final visualization, is done in Python. The frontend of the visualization tool that deploys the web application uses HTML, CSS and JavaScript.

As explained in Chapter 5, the Brat2Viz tool is organized into two modules: Brat2DRS and DRS2Viz. We will now describe the new icon visualization's implementation. The `Dictionary` and `Icon` classes were created in the DRS2Viz module, with the class `Dictionary` implementing the methods necessary to manipulate the objects in a way to automatically build the icon dictionary and use the information stored in it to compose the visualization.

For each integrated Application Programming Interface (API) and dataset there is a file with the functions that allow to search and handle the icons that correspond to the searched term. The `parser.py` file contains the functions created to process the information collected from the Discourse Representation Structure (DRS) regarding the selected news story. From fetching the

[1] Available at: https://github.com/LIAAD/Text2Icons

actors present in the news story, translating them and running the algorithm to obtain their more specific descriptions, to fetching information regarding the icons of the actors to be represented.

The HTML files that make up the visual part of the tool are located in the `templates` folder, where the file `icons.html` that builds the icon visualization was added. The `app.py` file gathers all the information needed to launch each visualization, starting a Flask[2] server running them.

For a user, the project's README file suffices to provide the necessary steps to get the tool up and running.

7.2 Icon Dictionary

The Icon Dictionary is built with the class `Dictionary`. Its methods allow one to search for icons to use in the visualization step of the pipeline. In this section, we discuss the dictionary construction, which sources of icons have been integrated and which remained as sources to use, which information about the icons and how it is saved, the two ways of choosing the icons to be added to the dictionary, and the supported features.

7.2.1 Icon Sources

Of the sources of icons explored, detailed in Chapter 4, the ones listed below are those that were integrated in the dictionary and with which it was possible to work to obtain and visualize icons. Of those mentioned here, only most of them are functional in the dictionary, for reasons to be explained next.

It is important to mention that in order to use the mentioned APIs it is necessary to have an account to obtain an API key, username or an authentication token, for each one of them. With regards to the datasets, Icons-50 does not need any type of authentication, whereas ImageNet needs a login account to access the data.

From APIs:

- **emojidex**[3]: it works perfectly, it searches using a term, returns the results and saves the icons without any problems, being one of the sources that remained working in the dictionary. The icon images are retrieved via an API `url` and saved in Portable Network Graphics (PNG) format;
- **IconFinder**[4]: similar to the case of emojidex, this one is also working and saves the icons in the same way;

[2]https://flask.palletsprojects.com/en/2.0.x/
[3]https://developer.emojidex.com/#api
[4]https://developer.iconfinder.com/reference/overview-1

- **Icons8**[5]: has been successfully integrated, and is working without problems. To support saving icons from this API, the images are converted from Scalable Vector Graphics (SVG) format to PNG;

- **OpenEmoji**[6]: searches for icons based on a term, returns a list of matched results, but it was not possible to save the icons as images, as it is an API that has the images in Unicode format.

From datasets:

- **Icons-50**[7]: search done successfully, this being another one of the icon sources in operation, with the images being saved in Joint Photographic Experts Group (JPEG) format;

- **ImageNet** - using Tiny Imagenet (Stanford CS231N)[8]: although this source worked as expected and was saving the images in JPEG format, it was not included in the sources to be used by the dictionary, for the same reason mentioned in Section 4.8, this dataset is composed by images rather than icons.

It is important to note that the quality and availability of the icons in the dictionary depends on the sources included in it, always having the possibility of adding new ones.

To sum up, of the icon sources that are integrated in the dictionary, the ones that are operational to search and use icons in the icon visualization are: emojidex, IconFinder, Icons8 and Icons-50.

7.2.2 Icon Information

The file `icon_info.json` stores the following information about the icons:

- `keyterm` of the icon, obtained from the name it has in the icon source of its origin;

- `variants`, list of associated/synonymous terms. This list is obtained using the method `findSyn(term)` that was composed with methods from the WordNet Interface[9], like `wordnet.synsets()` and `lemma_names()`;

- `img`, the file name of the new saved icon image;

- `source`, indicating the source where the icon was fetched;

- `icon_type`, indicates what type of icon was saved in relation to the searched term, in this case '`most`' for the most similar icon found.

[5] https://developers.icons8.com/docs/getting-started
[6] https://emoji-api.com/#documentation
[7] https://www.kaggle.com/danhendrycks/icons50
[8] http://image-net.org/
[9] http://www.nltk.org/howto/wordnet.html

Each time a new icon is added to the dictionary, a new entry in the `json` file is created with the corresponding icon data, and the icon image is saved under `drs2viz/static/icon_images`.

7.2.3 Dictionary Construction

Icons are searched by terms, adding a new icon when that term does not yet have an icon that represents it in the dictionary. When adding a new icon, a search is made on the available sources, with one of the icons in that list being saved. Searching for icons to populate the dictionary can be done in two ways: semi-automatically or automatically. These two available construction types have been implemented in the DRS2Viz module and are presented next.

7.2.3.1 Semi-automatic Construction

The main idea of using the semi-automatic construction to choose the icon that will be saved in the dictionary is to give the user the possibility to choose the icon from the list of results presented by the icon sources.

To semi-automatically search for an icon with a term, for instance with the term "computer", it is necessary to use the `search2Add(term,type,icon_type)` method as follows:

```
from Icon import *
from Dictionary import *

d = Dictionary()
d.search2Add('computer','semi','most')
```

Listing 7.1: Semi-automatic search example for an icon with a term.

The parameter `most` is the type of icon to search for. In this case, this method argument is not used, as it is a semi-automatic construction, and the choice of the icon is up to the user.

In Figure 7.1, we can see an illustrative example of the semi-automatic dictionary construction process. After calling the function, a list is returned for each icon source integrated in the dictionary. Following the Figure 7.1 enumeration, we have:

1. The list returned by the IconFinder API;
2. From the list elements, the user chooses one, giving it as input;
3. A new icon is then automatically saved in the dictionary, updating the `json` file, under `drs2viz/icon_info.json`, with the information regarding the new icon;
4. Saving the image of the new icon in the `drs2viz/static/icon_images` folder.

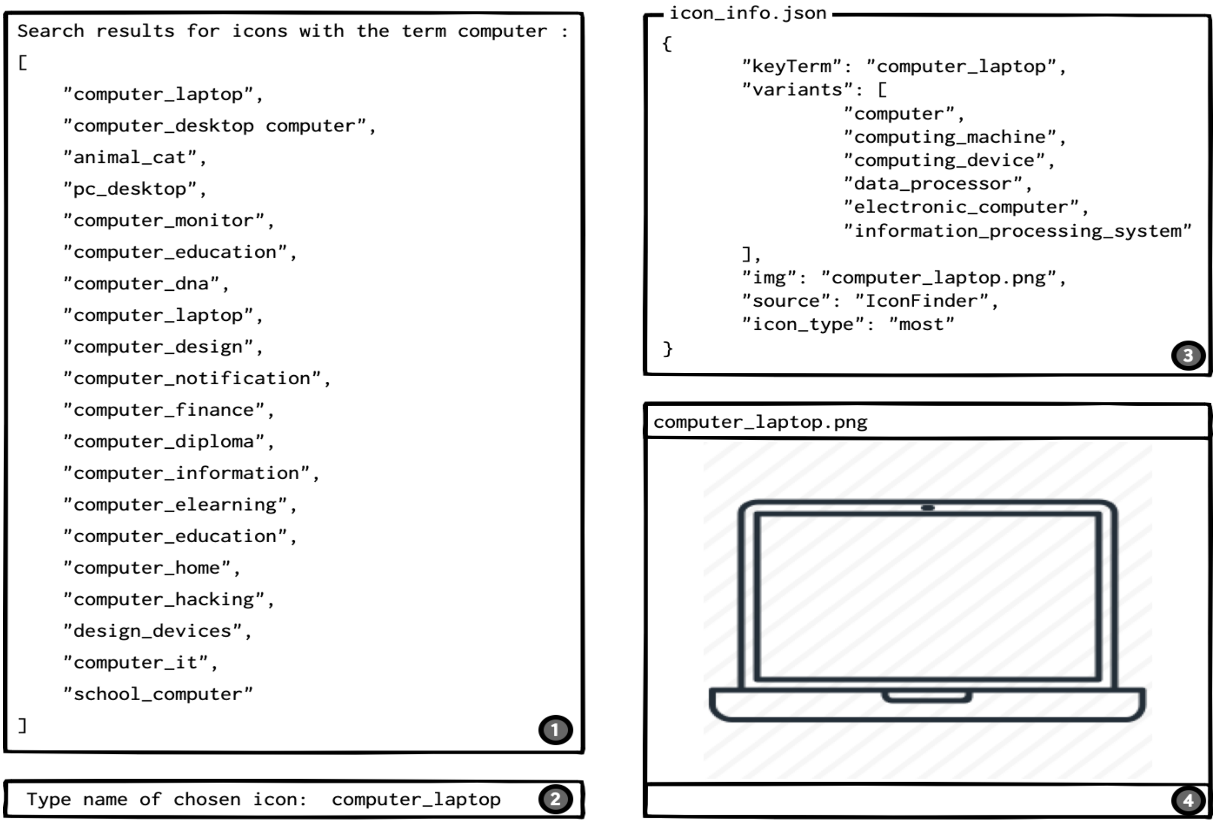


Figure 7.1: Example of semi-automatically adding an icon to the dictionary.

7.2.3.2 Automatic Construction

In order to automate the selection process, i.e., without the need for human intervention in choosing the icon to save, fastText[10] was used. This library for learning word embeddings and text classification allowed making the choice using models to obtain embeddings for any word. From the icon lists obtained from the searches in the icon sources, the cosine similarity between the searched term and each element of the lists is calculated to choose the icon that is most similar to the one searched for, so that it can be saved in the icon dictionary.

The search for the most similar icon was performed by calculating the cosine similarity between two word embeddings. This is done using the model `wiki-news-300d-1M-subword.bin`[11] and calling the method `get_word_vector()` to obtain the word embeddings, while using subword level information to build vectors for unknown words, i.e., getting an embedding for any word. To calculate the cosine similarity, the `scipy.spatial.distance.cosine()`[12] function was used.

To run an automatic search for an icon with a term, for instance with the term 'car', it is necessary to use the `search2Add(term,type,icon_type)` method. Similar to the semi-

[10] https://fasttext.cc
[11] https://dl.fbaipublicfiles.com/fasttext/vectors-english/wiki-news-300d-1M-subword.bin.zip
[12] https://docs.scipy.org/doc/scipy/reference/generated/scipy.spatial.distance.cosine.html

automatic search, but by changing the value of the `type` argument, the search type is immediately changed by using fastText to select the icon to be saved, choosing the one most similar to the search term, as shown below:

```
from Icon import *
from Dictionary import *

d = Dictionary()
d.search2Add('car','auto','most')
```

Listing 7.2: Automatic search example for an icon with a term.

7.2.4 Search for Opposite Icons

For evaluation purposes, further explained in Subsection 7.4.1, a dictionary of opposite icons was built automatically. The information is saved under `drs2viz/icon_opposite_info.json`, and the opposite icon images are saved under `drs2viz/static/opposites_images`, all integrated in the DRS2Viz module.

The `icon_opposite_info.json` file stores the following information about the opposite icons:

- `term`, search term for which an opposite icon is searched;
- `keyterm`, keyterm of the opposite icon;
- `img`, the file name of the opposite icon image;
- `source`, indicates the source where the opposite icon was fetched;
- `icon_type`, indicates what type of icon was saved in relation to the search term ('`least`' for the least similar or '`2most`' for the second most similar).

To visualize the news stories with the opposite icons it is necessary to comment and uncomment a few code lines in the files: `drs2viz/parser.py` and `drs2viz/templates/icons.html` (described in the respective files what to do).

7.2.5 Features

The class `Dictionary` allows the manipulation of terms and icons using a set of implemented methods. Among which are:

- `updateIcons()` - updates the icons from the icon image folder to the dictionary, being used when saving a new icon;

- `getDictionary()` - lists all icon terms present in the dictionary;
- `showJson()` - shows the icons information present in the JSON file;
- `iconInfo(term)` - shows the icon information of a term;
- `alternativeIcons(term)` - shows the alternative icons of a term;
- `showIcon(term)` - displays an icon image in a display window, if the term exists in the dictionary. In that case, it uses the `displayIcon()` method defined in the `Icon` class;
- `search(term)` - checks if `term` is in the dictionary. When it does not find an icon, it looks in the variants of that term;
- `searchImgName(term)` - gets the icon image's name for `term` by searching for the most similar in the dictionary using fastText. It is used for the visualization in the web application;
- `updateJson(keyTerm,variants,img,source,icon_type)` - updates the JSON file with the icon information when saving a new one;
- `search2Add(term,type,icon_type)` - dynamically adds an icon to the dictionary, and when there is no icon for a specific term, searches the available sources, allowing to choose the type of search with the argument `type` (semi-automatic, automatic or opposite).

For the opposites:

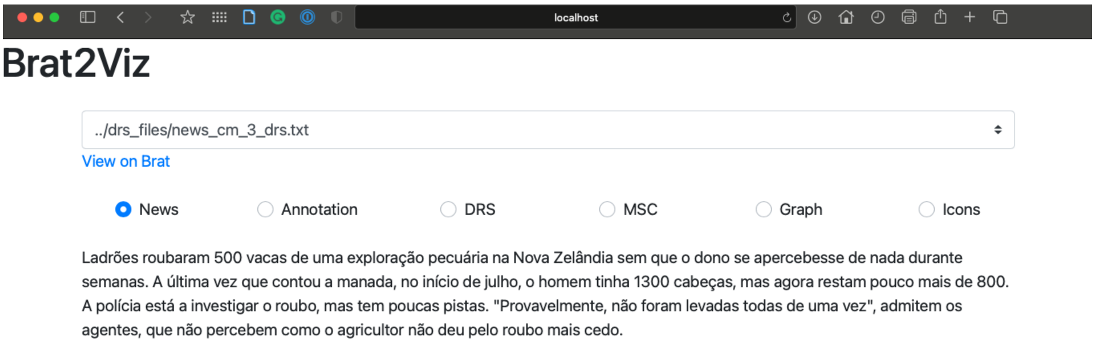
- `updateOpposites()` - similar to `updateIcons()`;
- `searchOp(term,icon_type)` - similar to `search(term)`;
- `searchImgNameOp(term,icon_type)` - similar to `searchImgName(term)`;
- `updateJsonOp(term,keyTerm,img,source,icon_type)` - similar to `updateJson()`.

7.3 Visualization Deployment

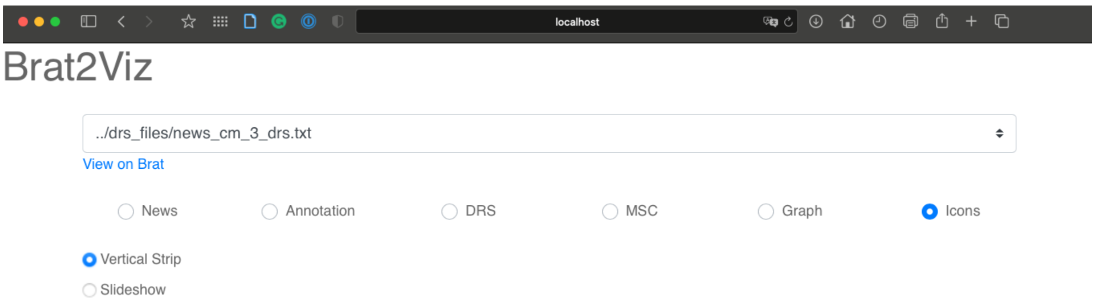
Now with the sentences extracted, the actors identified, translated and their most specific descriptions identified by the algorithm, the icon dictionary built and organized, and with the search for icons done in an automated way, it was necessary to link everything so that, with the information collected and processed from the news story, we could see the icon visualization as the last step of the pipeline.

In Figure 7.2, are presented some of the options available for the user to select in order to, for example, see the selected news story text, in 7.2a, and to observe the icon visualization, in 7.2b.

It is important to note that, by default, the translation method being used to obtain the final icon visualization is Googletrans, and the dictionary search and its construction are being done automatically using fastText by always searching for the icon most similar to the desired search.



(a) Selection of the news story text option.



(b) Selection of the icon visualization option.

Figure 7.2: Overview of some of the Brat2Viz narrative visualization tool options, corresponding to Example (5).

The integration of the icon dictionary was done in the DRS2Viz module in order to deploy the web application with the generated icon visualization. To compose the visualization, two types of visualization using icons were created: vertical strip and slideshow. It is possible to choose the visualization option for the selected news story. To exemplify the types of visualization implemented, consider the news story in Example (5), where the actors are highlighted in blue:

(5) Ladrões roubaram 500 vacas de uma exploração pecuária na Nova Zelândia sem que o dono se apercebesse de nada durante semanas. A última vez que contou a manada, no início de julho, o homem tinha 1300 cabeças, mas agora restam pouco mais de 800. A polícia está a investigar o roubo, mas tem poucas pistas. "Provavelmente, não foram levadas todas de uma vez", admitem os agentes, que não percebem como o agricultor não deu pelo roubo mais cedo.

In Table 7.1 are exemplified the processing steps through which the information goes through in order to build the icon visualization for Example (5):

Dictionary of the actors present in the news story	{'T1':'Ladrões', 'T3':'500 vacas, todas', 'T4':'uma exploração pecuária', 'T5':'Nova Zelândia', 'T6':'o dono, o homem, o agricultor', 'T8':'durante semanas', 'T10':'a manada, 1300 cabeças', 'T11':'início de julho', 'T14':'pouco mais de 800', 'T16':'A polícia, os agentes', 'T18':'o roubo, roubo', 'T20':'poucas pistas', 'T30':'agora'}
Translation of the actors Dictionary	{'T1':'Thieves', 'T3':'500 cows, all', 'T4':'a livestock farm', 'T5':'New Zealand', 'T6':'the owner, the man, the farmer', 'T8':'for weeks', 'T10':'the herd, 1300 heads', 'T11':'early july', 'T14':'just over 800', 'T16':'The police, officers', 'T18':'the theft, theft', 'T20':'few clues', 'T30':'now'}
Algorithm result (most specific descriptions)	{'T1':'thief', 'T3':'cow', 'T4':'farm', 'T5':'zealand', 'T6':'man', 'T8':'week', 'T10':'herd', 'T11':'july', 'T14':'', 'T16':'police', 'T18':'theft', 'T20':'clue', 'T30':''}
Icon files that match the terms (search for the most similar icon)	['thief.png', 'cow.png', 'farm_farming.png', 'nz_flag.png', 'man.png', 'appointment_calendar.png', 'farm_herd.png', 'calendar_jul.png', 'car_police.png', 'car_theft.png', 'clue_evidence.png']
The icons that appear in each sentence	{'S1':['thief.png', 'cow.png', 'farm_farming.png', 'nz_flag.png', 'man.png', 'appointment_calendar.png'], 'S2':['man.png', 'farm_herd.png', 'calendar_jul.png'], 'S3':['car_police.png', 'car_theft.png', 'clue_evidence.png'], 'S4':['cow.png', 'man.png', 'car_police.png', 'car_theft.png']}

Table 7.1: Explanation of the information processing to build the icon visualization for Example (5).

7.3.1 Vertical Strip

The idea behind the vertical strip is to present the full text of the news story so that there is an overview of the entire story with the icons that represent it. The text of the news story is presented separately sentence by sentence and the respective actors of each sentence are represented with icons. An example of a vertical strip generated for a news story can be seen in Figure 7.3.

Figure 7.3: Vertical strip visualization of Example (5).

7.3.2 Slideshow

The goal of the slideshow option is to allow the user to interact with the visualization. This visualization option was based on a template[13] from CodePen[14]. In Figure 7.4 is shown one of the slides of the complete slideshow generated for Example (5). As we can see, the user can interact with the visualization by clicking on the left arrow, which makes the slideshow move to the slide of the previous sentence in the text, and by clicking on the arrow on the right, it moves to the slide of the next sentence.

[13] https://codepen.io/andytran/pen/xweoPN/
[14] https://codepen.io

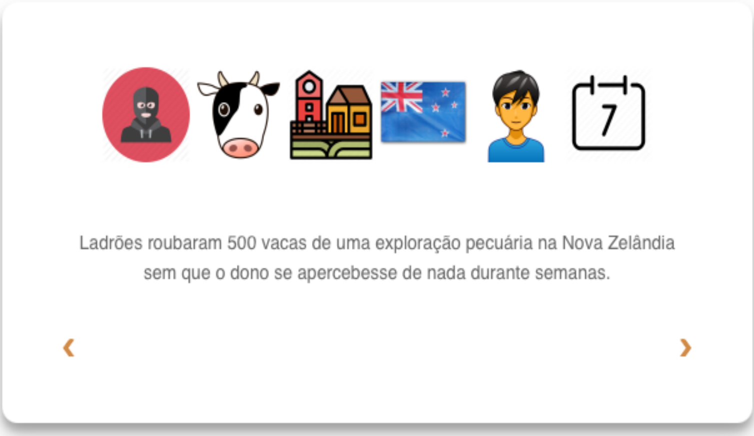


Figure 7.4: Slideshow visualization of Example (5).

7.4 Analysis and Evaluation

With the implementation of the icon visualization completed, it became important to understand if the results obtained in the visualization are adequate, i.e., if the generated visualizations represent the stories efficiently from the perspective of a potential user.

In order to assess both the quality of the resulting news stories representations, and the correctness and efficiency of matching terms with representative icons, two surveys were carried out. One focused on stories to measure the ability of a set of icons to represent a scene in a story, evaluating the final result. The other is a survey focused on icons to measure the quality of the term-icon connection, evaluating the intermediate result. They are both presented in Subsection 7.4.1 and Subsection 7.4.2, respectively. We also found it relevant to try to understand people's views on the use of illustrative icons as a complement to news stories, showing the respective results in Subsection 7.4.3.

The answers obtained from the surveys were collected by a network of direct and indirect personal contacts, and by the academic community of the University of Porto.

7.4.1 Visualization of News Stories

In order to evaluate the ability of a set of icons to represent a story, taking into account the connection of the actors in the story to a particular icon that represents it, survey participants were asked to rate from 1 to 5 how well a set of icons represented the sentence in question, with 1 being "Completely disagree" and 5 being "Completely agree". For this survey, responses were collected from 149 participants. The full survey is presented in Appendix A.

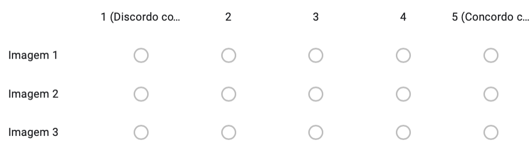
Figure 7.5 shows the question presented for the setence "Ladrões roubaram 500 vacas de uma exploração pecuária na Nova Zelândia sem que o dono se apercebesse de nada durante semanas.", with the images that the participants have to analyze (7.5a, 7.5b, and 7.5c), and the image voting scheme (7.5d).

The survey features 10 questions, each with a different sentence. The sentences were taken from the project's news stories collection, with no more than two sentences belonging to the same news story. Each question has 3 images associated with a set of icons, one of these sets corresponds to the sequence of icons generated by the visualization when the survey was built. The other two sets were automatically generated by the visualization, but instead of searching for the icons that are most similar to the searched term, one of the sets displays the second most similar icons, and the other set displays the least similar ones. The implementation made to automatically search for opposite icons in order to get the icons that make up these sets, the second most similar and the least similar, is integrated in the DRS2VIZ module, having the possibility to automatically build a dictionary of opposite icons.


(a) Image 1.

(b) Image 2.


(c) Image 3.



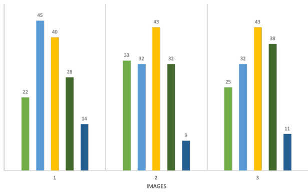
(d) Form for participant responses.

Figure 7.5: Survey options for the sentence "Ladrões roubaram 500 vacas de uma exploração pecuária na Nova Zelândia sem que o dono se apercebesse de nada durante semanas."

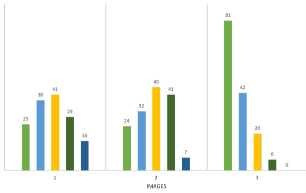
To analyze the distribution of responses to the survey questions, we present the histograms in Figure 7.6, in which we can see that the dispersion of responses is relatively high, with some cases with greater agreement among the participants. The analysis carried out for this survey also included the calculation of the average of the scores given by the participants to each image of each question. This is so later we can compare if the highest score of a question corresponds to the image created by the visualization with the most similar icons. Thus, we present in Table 7.2 the mean scores of each image for the 10 questions of the survey. The cells that correspond to the image generated for the most similar icons are displayed with color: green for the images with the highest score; yellow for the ones with the second best score; and orange for the worst score.

Questions	Images		
1	2.779	2.678	2.852
2	2.819	2.832	1.671
3	1.617	2.758	1.906
4	3.107	2.128	2.342
5	1.859	3.530	2.470
6	1.946	1.819	2.242
7	2.376	3.235	3.107
8	2.846	2.000	1.913
9	2.201	2.409	2.846
10	2.027	3.570	1.899

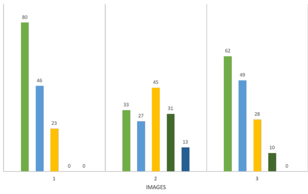
Table 7.2: Mean scores of each image for the survey questions.



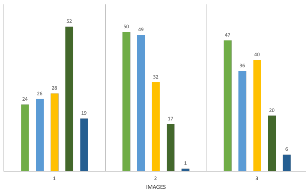
(a) Question 1.



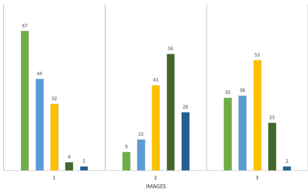
(b) Question 2.



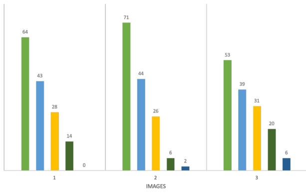
(c) Question 3.



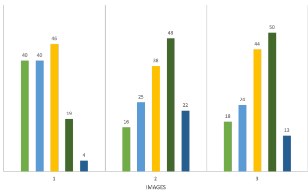
(d) Question 4.



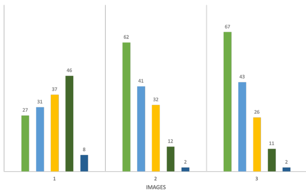
(e) Question 5.



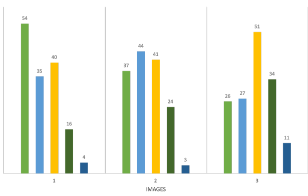
(f) Question 6.



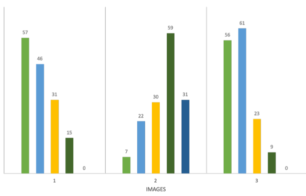
(g) Question 7.



(h) Question 8.



(i) Question 9.



(j) Question 10.

(k) Legend.

Figure 7.6: Distribution of responses to each question in the survey.

Although the results indicate that most of the answers are divided, it can be observed that the average of the scores is more expressive with regards to the answer with the best score corresponding to the image generated with the most similar icons. Looking at the histograms, we see that there is a group of participants who do not agree with the image composed by the most similar icons, but even so, there is a large number that do.

The image with the set of icons with the highest score, i.e., the most scored on average by the participants, corresponds to the one generated by the visualization with the most similar icons in 80% of the cases. This value is very favorable for the purpose of analyzing the data in this survey, as it indicates that possible news readers or users of the visualization tool identify the majority of the icon visualizations generated for the sentences presented as the most suitable. Thus concluding with this analysis that the representation of the story as a whole is in accordance with what users expect to see.

7.4.2 Term-Icon Connection

To assess the quality of linking a term to an icon that represents it, survey participants were asked to choose the icon that best suited the term presented, from a set of 6 icons, as we can see in Figure 7.7, which shows the options available for the term “Man”. The survey has a total of 20 questions, each with one term, and it can be seen in its entirety in Appendix B. The list of terms can be seen in the first column of Table 7.3. Among the 6 icons presented for each term, one of them corresponds to the icon selected by the visualization as the most similar to the searched term. The remaining icons also belong to the icon dictionary, having been chosen because they are possible representations of the term in question.

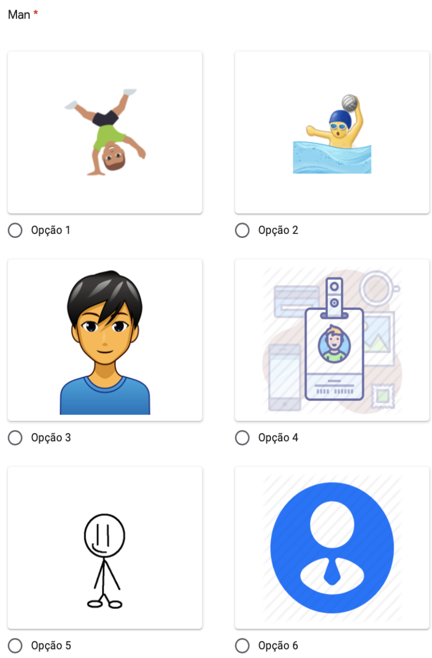


Figure 7.7: Survey options for the term “Man”.

As a starting point in the analysis of the data obtained through the survey responses, the Fleiss’ kappa [40] was calculated as a preliminary evaluation to understand if the answers were chosen randomly or not, measuring their reliability. The **kappa** statistic is a measure of agreement between raters/observers. In 1960, Cohen introduced a measure of nominal scale agreement [22], and in 1968, extended this measure to a weighted kappa statistic [23]. Both were defined for only two raters. Fleiss introduced an unweighted kappa statistic, κ, for 2 or more raters. Within the scope of this analysis, the raters are the participants who responded to the survey, and since there were more than 2 participants, Fleiss' kappa was the chosen measure to apply to the collected data.

To proceed with the calculation of κ, the following notation was used:

- N number of cases/questions = 20 questions;
- n number of raters/survey participants = 291 participants;
- k number of options in which a case can be rated = 6 icons presented for each.

This definition is based on an N by k observation table or matrix, where the elements n_{ij} represent the number of participants who assigned the i-th term to the j-th icon. Therefore, following Fleiss's notation [40]:

$$p_j = \frac{1}{Nn}\sum_{i=1}^{N} n_{ij}, \tag{7.1}$$

where p_j is the proportion of all assignments to the j-th icon,

$$P_i = \frac{1}{n(n-1)}\sum_{j=1}^{k} n_{ij}(n_{ij}-1) = \frac{1}{n(n-1)}\left(\sum_{j=1}^{k} n_{ij}^2 - n\right), \tag{7.2}$$

P_i is the extent of agreement among the n participants for the i-th term,

$$\bar{P} = \frac{1}{N}\sum_{i=1}^{N} P_i, \tag{7.3}$$

$\bar{P}$ is the observed overall agreement,

$$\bar{P}_e = \sum_{j=1}^{k} p_j^2, \tag{7.4}$$

and P_e is the expected mean proportion of agreement due to chance, i.e., what would be expected if participants made their choices at random.

With $1 - \bar{P}_e$ measuring the degree of agreement attainable above chance, and the degree of agreement that is actually attained in excess of chance being $\bar{P} - \bar{P}_e$, the kappa statistic is defined as follows:

$$\kappa = \frac{\bar{P} - \bar{P}_e}{1 - \bar{P}_e}. \tag{7.5}$$

For the data collected by the survey, shown in Table 7.3, we obtain the following results:

$$\begin{aligned} \bar{P} &= 0.486574239 \\ \bar{P}_e &= 0.173819334 \\ \kappa &= 0.3785551 \end{aligned}$$

	Icons (j)						
Term (i)	1	2	3	4	5	6	P_i
Man	3	1	175	15	21	76	0.43590473
Police	136	148	0	1	6	0	0.47571987
Car	6	0	6	1	1	277	0.90664771
Crime	6	48	140	1	67	29	0.31970613
Thief	8	39	202	2	32	8	0.5117905
Gun	1	2	6	281	1	0	0.93271715
Attack	127	5	40	6	101	12	0.32994431
Woman	4	1	0	277	8	1	0.90674251
School	83	5	61	15	54	73	0.22294111
Business	152	82	19	2	9	27	0.36392938
Plant	0	75	0	4	184	28	0.47387131
Imagination	105	15	1	0	161	9	0.43799028
Fight	0	55	0	188	4	44	0.4743453
Fruit	0	17	59	116	99	0	0.31681479
Combat	1	8	1	181	56	44	0.44564522
Rest	4	65	1	13	1	207	0.55658253
Farm	46	120	26	30	49	20	0.24412845
Europe	140	8	99	7	0	37	0.36250741
Fire	1	1	0	251	37	1	0.75935537
Conference	25	59	106	0	76	25	0.25420073
Total	848	754	942	1391	967	918	
p_j	0.14570447	0.12955326	0.16185567	0.23900344	0.1661512	0.15773196	

Table 7.3: Data obtained from the responses to the Term-Icon survey.

According to the kappa interpretation values presented by Landis et. al. [66], showed in Table 7.4, the obtained kappa value is classified as Fair agreement. With the value being very close to the next range of values, we conclude that the kappa value is reliable, eliminating the possibility that the questions were answered completely at random, making the data collected by the survey suitable for further analysis.

κ	**Strength of Agreement**
<0.00	Poor
0.00 – 0.20	Slight
0.21 – 0.40	Fair
0.41 – 0.60	Moderate
0.61 – 0.80	Substantial
0.81 – 1.00	Almost perfect

Table 7.4: Interpretation of kappa values.

The next step was to calculate the hit rate of each question, in order to see if the participants' choices match the icon selected by the visualization as the most suitable for the representation. The hit rate is calculated as follows:

$$HitRate = \frac{\#Hits}{\#Hits + \#Misses} \tag{7.6}$$

We consider that a **hit** is when the chosen icon is the one selected by the visualization, and a **miss**, otherwise.

The data presented in Figure 7.8 corresponds to the hit rates for each term of the survey.

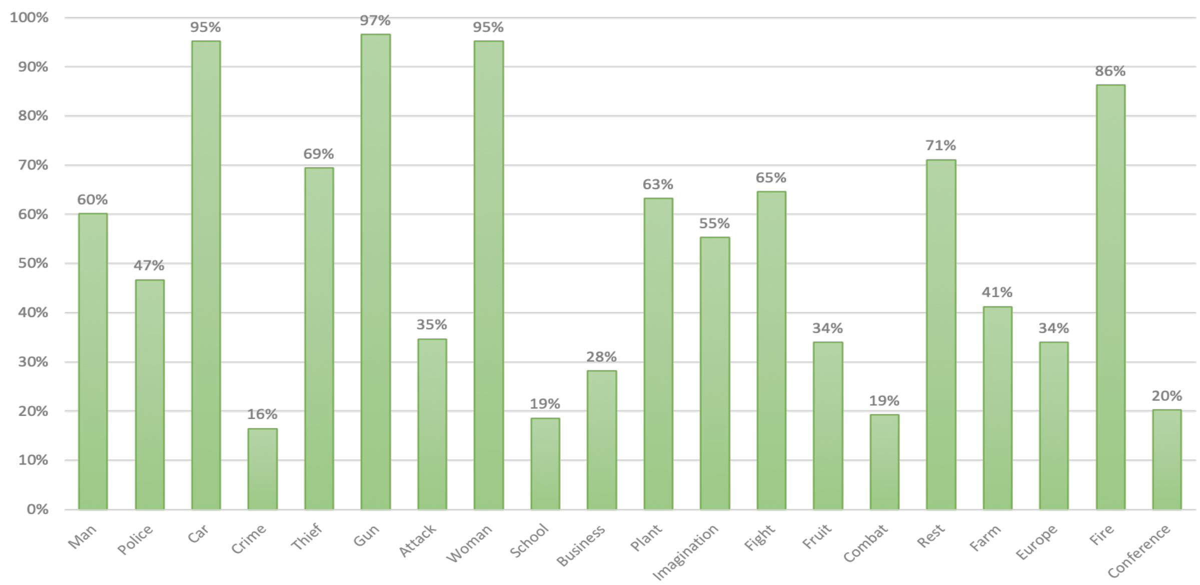


Figure 7.8: Hit rate of survey responses for each term.

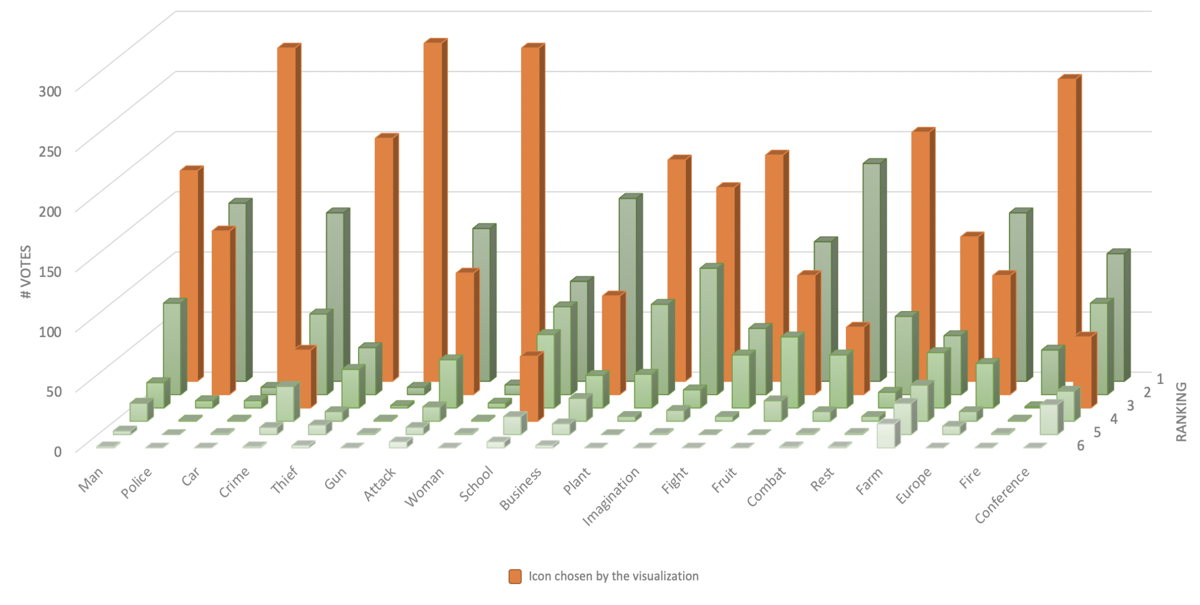


Figure 7.9: Ranking of survey responses for each term.

In Figure 7.9, an overview of the distribution of responses to the survey of terms is presented, observing the ranking (from the most voted to the least voted option), where the option that corresponds to the one chosen by the visualization is highlighted in orange.

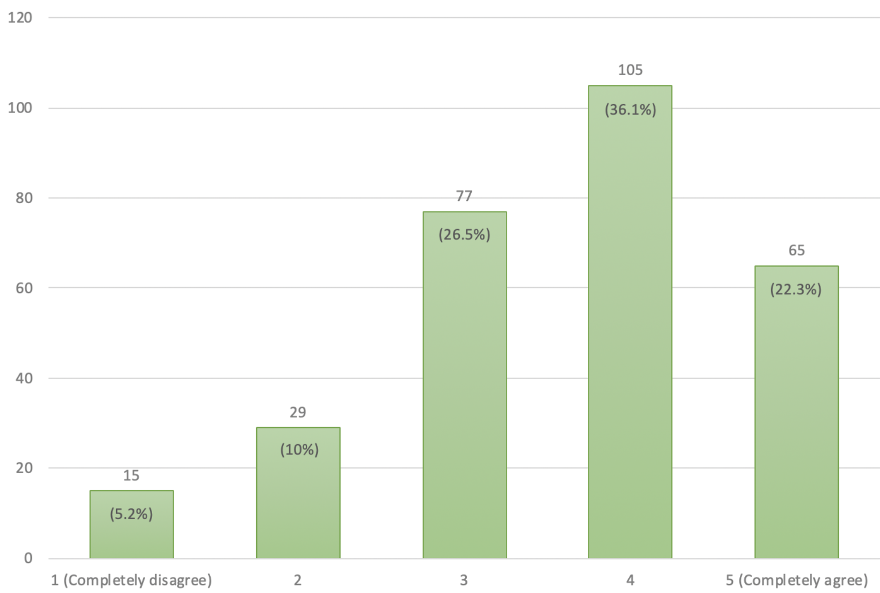
The average hit rate for each question with a term is 53%. It is important to note that, for the entire survey, the percentage for the most voted icon matching the icon chosen by the visualization is 55%, with 30% for the second most voted, 10% for the third, and 5% for the fourth, with the fifth and sixth with 0%. This results in 85% of the cases falling into the first and second most voted option, so we concluded that the results are favorable to our assessment, since the visualization presents icons that users identify as the most suitable for the terms presented.

7.4.3 Opinion of survey participants

The opinion of possible news readers and users of the visualization is crucial to understand the scope of the visualization presented here, how it would be received and what importance would be given to it. In order to have a better understanding of this issue, two opinion questions were outlined. Participants were asked to rate from 1 to 5, with 1 being "Completely disagree" and 5 "Completely agree", the following questions:

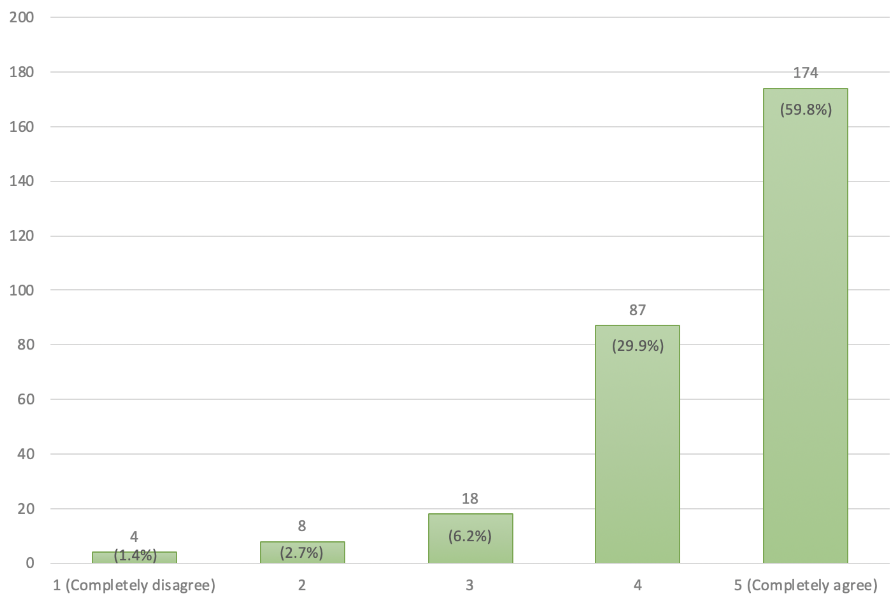
1. As a news reader, would you find it useful for you to have the text of some news accompanied by icons illustrating them?
2. Would you find it useful for other news readers (for example children or people with reading difficulties) that the text of some news was accompanied by icons illustrating them?

With a total of 291 responses, Figure 7.10a and Figure 7.9b show the distribution of responses obtained for Question 1 and Question 2 respectively.



(a) Distribution of answers to Question 1 about whether survey participants find it useful to themselves.

As we can see, for Question 1, more than half of the participants gave their answer to options 4 or 5, options with the highest level of agreement, gathering a total of 58.4% of answers in



(b) Distribution of answers to Question 2 about whether survey participants find it useful to children or people with reading disabilities.

Figure 7.9: Data obtained from the survey opinion questions.

favor of the usefulness they consider for themselves. For Question 2, the results are even more favorable, with 89.7% of the answers in options 4 or 5, concluding that the participants' point of view regarding the usefulness for children or people with reading disabilities reflects the premise here addressed: the importance of narrative visualization.

7.5 Summary

This chapter explains the last steps taken to obtain the icon visualization. From the construction and use of the icon dictionary, to the deployment of the visualization options, and finishing with a critical analysis of the results obtained from surveys carried out in order to evaluate the results of the proposed visualization.

Chapter 8

Conclusions

In this work, we proposed a new narrative visualization by representing key concepts of a narrative with icons. This visualization is integrated in a Narrative Annotation Visualization tool called Brat2Viz.

We started by introducing the research carried out related to narrative extraction by addressing some Natural Language Processing (NLP) tasks used for automatic identification and extraction of narrative elements, and by presenting an overview of neural language models and representation using embeddings. This research served to understand how narratives can be represented, how to automatically extract narrative elements to gather information about a story, and emphasized the value of information representation using embeddings. The topic of narrative visualization was also covered, highlighting its importance and addressing some current approaches. Then, the icon Application Programming Interfaces (APIs) and datasets that were researched were presented as possible sources of icons to be used in the course of the work, with an evaluation analysis regarding them also shown. In Chapter 5, we presented an overview of the demonstration pipeline, explaining the steps that compose it, making it clear where the new visualization fits with the Discourse Representation Structure (DRS) files providing the necessary news story information as input for the visualization, and the output being a set of icons that represents the story.

In Chapter 6, we introduced the necessary process to extract narrative elements, explaining the steps that later supported the construction of the visualization. Among them, the extraction of sentences from the initial text of the news stories, the integration of automatic translation methods (Hugging Face Transformers and Googletrans), and also the presentation of the Actor Level Resolution Algorithm that finds the most specific descriptions of the actors present in the news stories. Both the translation methods and the algorithm were evaluated and analyzed, being concluded that, in terms of performance and quality of the translation generated, the most favorable translation method is Googletrans. Regarding the algorithm, when using translations by Googletrans, it presents 67.89% of well defined actors. At this point of the work, some limitations were pointed out that may be possible starting points for future work. The fact that if an actor has a more complex description, being a person or a place, for example, it can lead to faulty descriptions in the result presented by the algorithm. The algorithm only deals with one-term descriptions, which is also a limitation, as it can lose its meaning as the algorithm progresses.

Chapter 7 begins by introducing the icon dictionary that was designed to be used as a database to generate the icon visualization. The explored icon sources that were found to be adequate were integrated in the dictionary, with the APIs emojidex, IconFinder and Icons8, and the Icons-50 dataset remaining operational. The dictionary has two types of construction available, that is, two ways of adding new icons to the dictionary: semi-automatically or automatically. The semi-automatic gives the user the possibility of choice, and the automatic saves the most similar icon provided by the sources. The icon visualization combined all these steps, which resulted in the creation of two visualization options: vertical strip and slideshow. The first presents the complete news story divided into sentences, with the actors of each sentence represented with icons. The second one presents cards, sentence by sentence, also showing the icons that correspond to the actors. This gives the user the possibility to interact with the visualization by moving to the previous or next sentence.

Finally, it was important to evaluate the results obtained from the point of view of potential users. Two surveys were organized to assess both the quality of the generated news stories representations, and the correctness and efficiency of matching terms with representative icons. Responses from 149 and 291 people were collected and analyzed for each survey, respectively, with very favorable results for the topic and motivation presented in this work. For the survey regarding the stories visualizations, the most voted image matches the generated visualization in 80% of cases. As for the term-icon survey, 85% of the cases where the most voted icon matches the icon chosen by the visualization fall either the first or second most voted option. Answers to two opinion questions were also collected, which reflected the importance of narrative visualization.

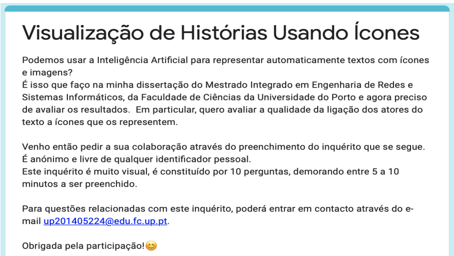
It might be interesting, as future work, to expand the visualization to automatically generated icons, i.e., generating icons on the fly instead of resorting to icon sources for the searched terms. It will be useful and important to incorporate the information from news sentences in the DRS itself. In addition to representing the actors, it can be useful to find appealing ways to represent the events of the narratives. With regards to the integration of new icon sources, the resolution of the use of Unicode characters is also relevant, and word-emoji embeddings [98] can also be explored as a linking tool between emojis and words. As already mentioned, the algorithm also has room for improvement when it comes to being able to handle more complex actors.

Ultimately, we conclude that, even with room for improvement both in information processing and in the final visualization, the work presented here achieves the proposed objectives: it introduces a new methodology for building icon dictionaries; performs automatic icon search; features a generality level resolution algorithm; and the new icon-based visualization method was successfully integrated into the project's pipeline, with good results, as verified by the analysis of the surveys carried out.

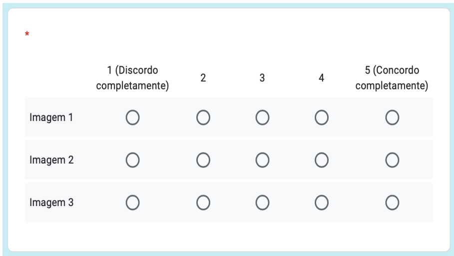
Appendix A

Survey: Visualization of News Stories

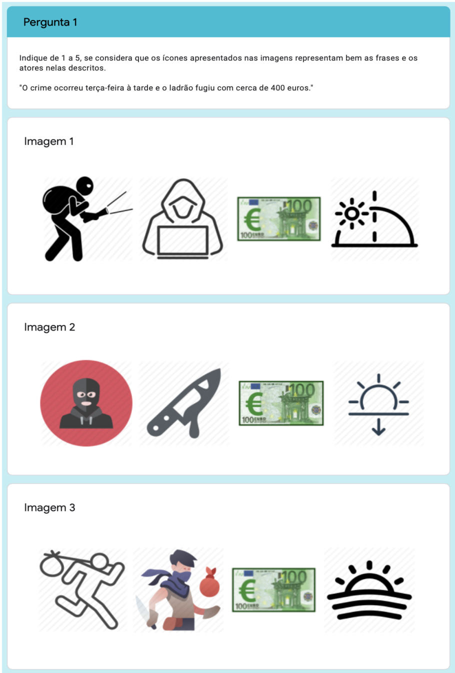
In this appendix the complete survey related to the evaluation of a set of icons representing a story, as discussed in Subsection 7.4.1, can be seen in Figure A.1.



(a) Introduction to the survey.



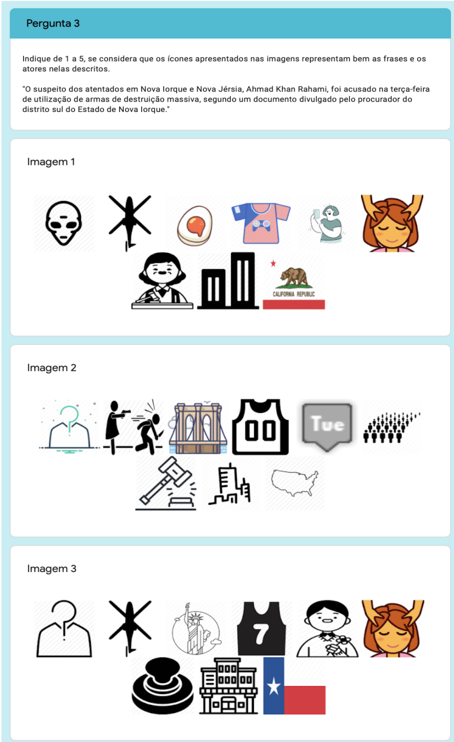
(b) Response scheme for each question.



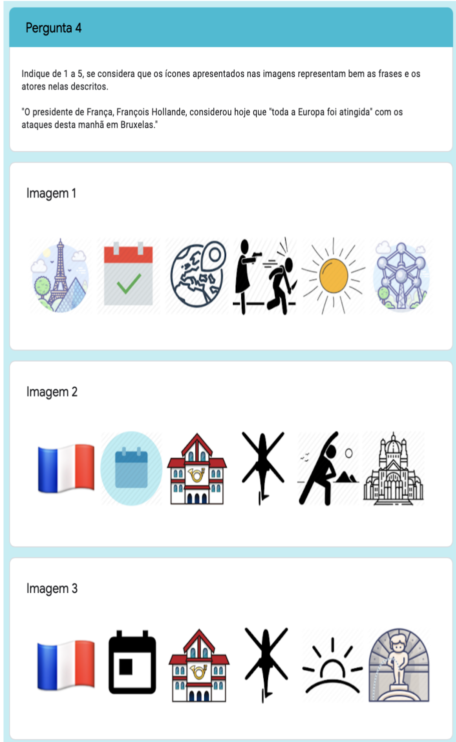
(c) Question 1.

(d) Question 2.

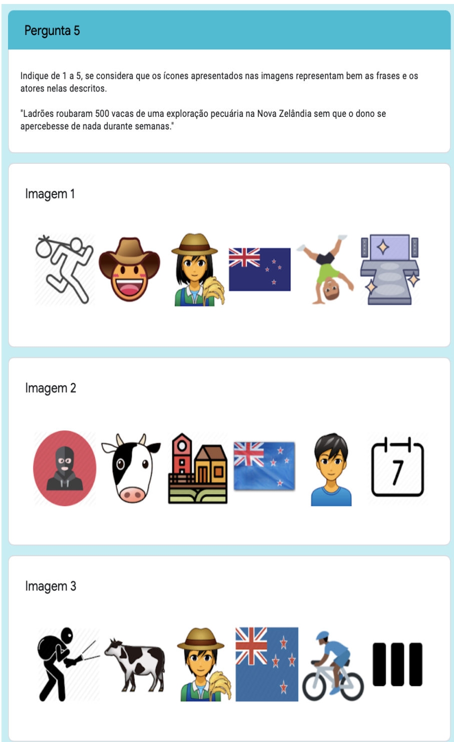
Figure A.1: Complete survey regarding the Visualization of News Stories.



(e) Question 3.



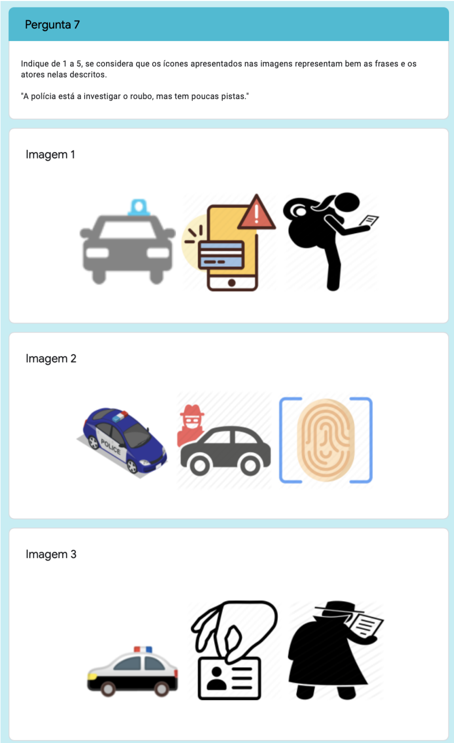
(f) Question 4.



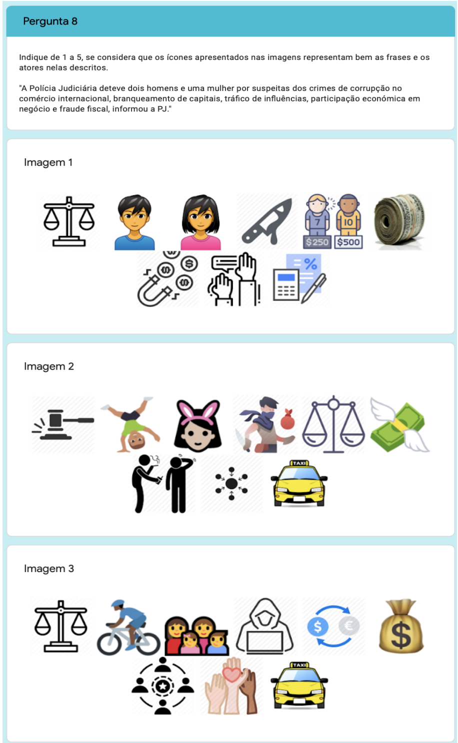
(g) Question 5.

(h) Question 6.

Figure A.1: Complete survey regarding the Visualization of News Stories.

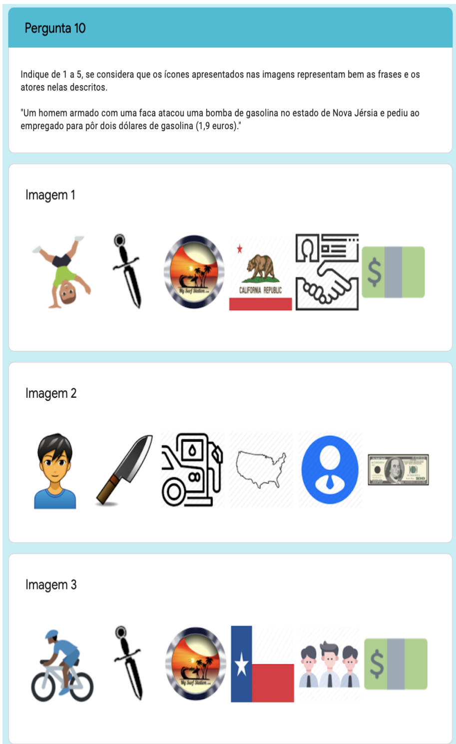


(i) Question 7.



(j) Question 8.

(k) Question 9.



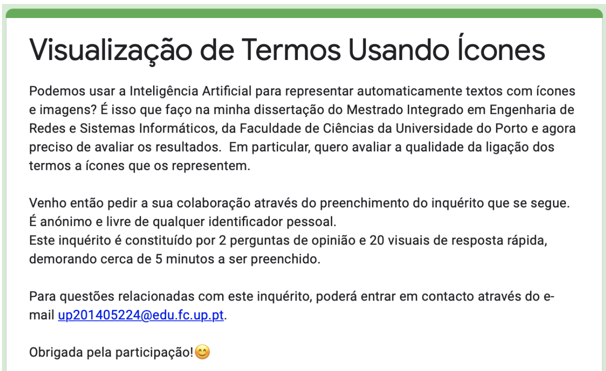
(l) Question 10.

Figure A.1: Complete survey regarding the Visualization of News Stories.

Appendix B

Survey: Term-Icon Connection

This appendix presents the complete survey in Figure B.1, concerning the evaluation of the term-icon connection, discussed in Subsection 7.4.2.



Visualização de Termos Usando Ícones

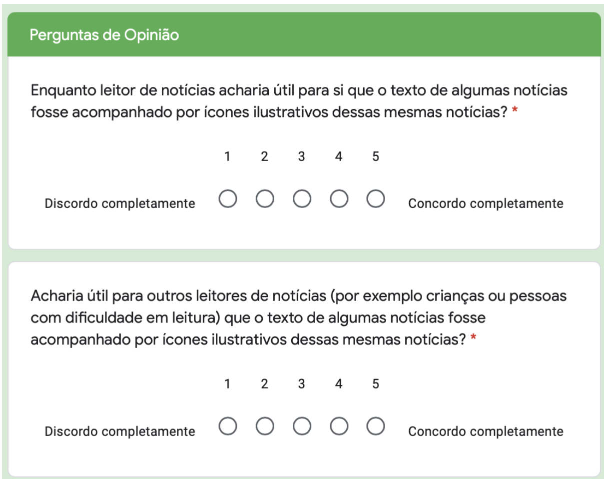
Podemos usar a Inteligência Artificial para representar automaticamente textos com ícones e imagens? É isso que faço na minha dissertação do Mestrado Integrado em Engenharia de Redes e Sistemas Informáticos, da Faculdade de Ciências da Universidade do Porto e agora preciso de avaliar os resultados. Em particular, quero avaliar a qualidade da ligação dos termos a ícones que os representem.

Venho então pedir a sua colaboração através do preenchimento do inquérito que se segue.
É anónimo e livre de qualquer identificador pessoal.
Este inquérito é constituído por 2 perguntas de opinião e 20 visuais de resposta rápida, demorando cerca de 5 minutos a ser preenchido.

Para questões relacionadas com este inquérito, poderá entrar em contacto através do e-mail up201405224@edu.fc.up.pt.

Obrigada pela participação!😊

(a) Introduction to the survey.



Perguntas de Opinião

Enquanto leitor de notícias acharia útil para si que o texto de algumas notícias fosse acompanhado por ícones ilustrativos dessas mesmas notícias? *

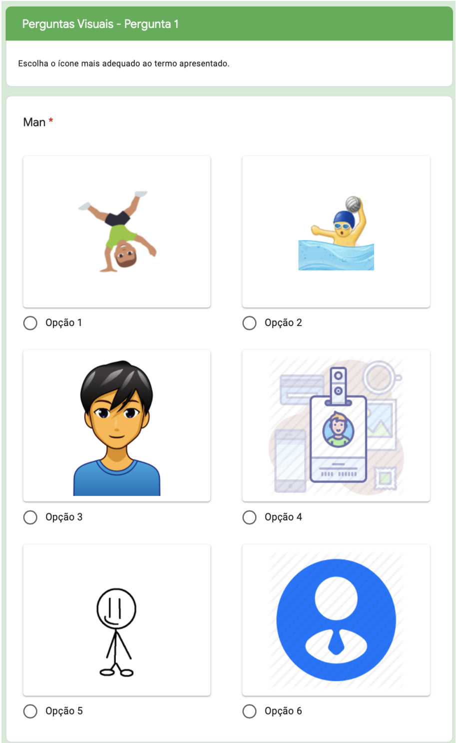
	1	2	3	4	5	
Discordo completamente	○	○	○	○	○	Concordo completamente

Acharia útil para outros leitores de notícias (por exemplo crianças ou pessoas com dificuldade em leitura) que o texto de algumas notícias fosse acompanhado por ícones ilustrativos dessas mesmas notícias? *

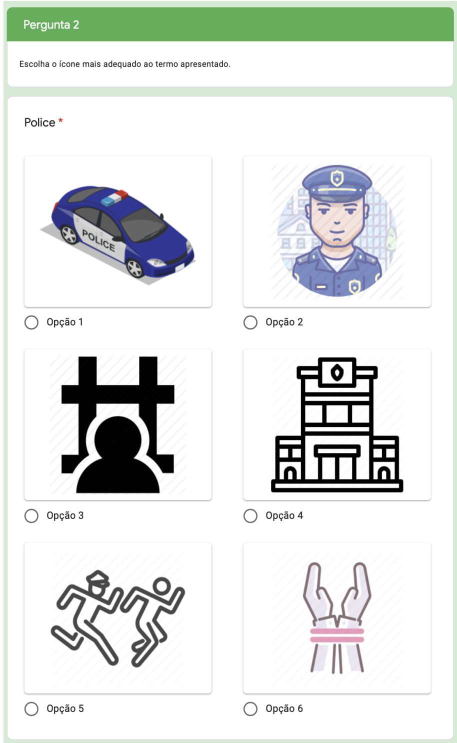
	1	2	3	4	5	
Discordo completamente	○	○	○	○	○	Concordo completamente

(b) Opinion questions.

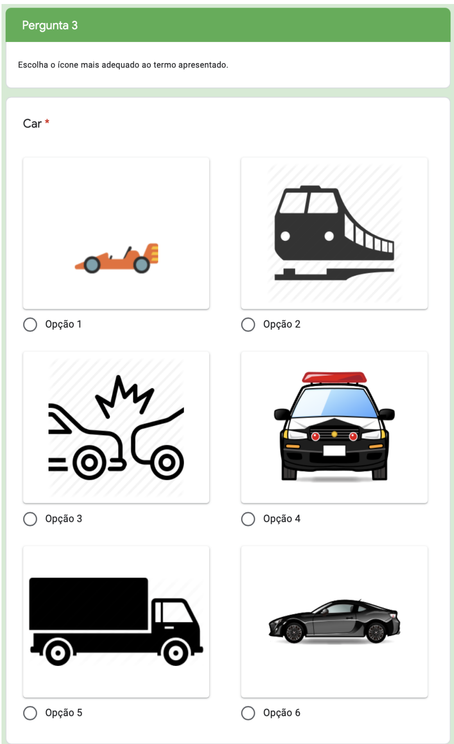
Figure B.1: Complete survey regarding the Term-Icon Connection.



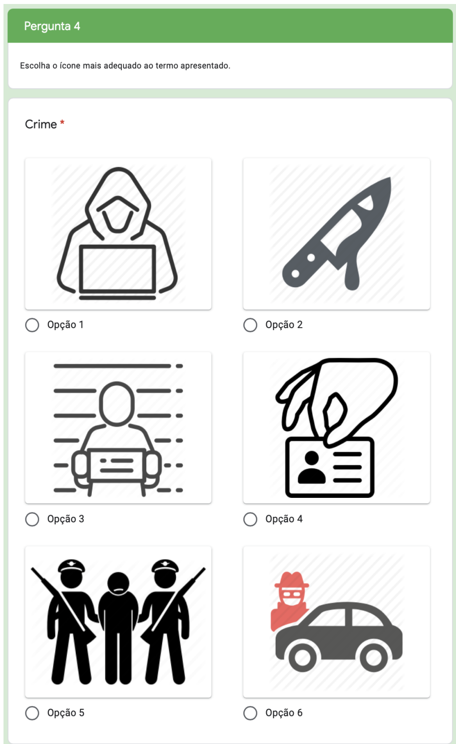
(c) Question 1.



(d) Question 2.

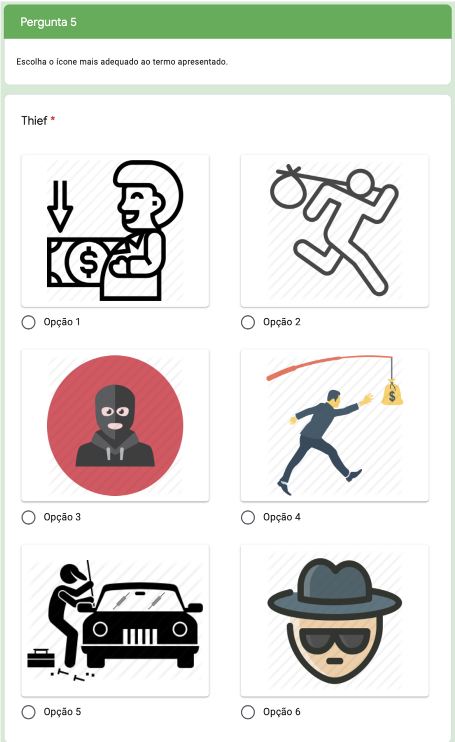


(e) Question 3.

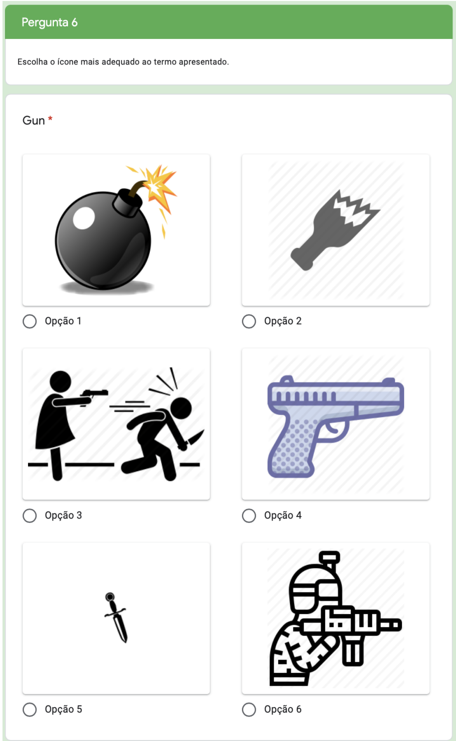


(f) Question 4.

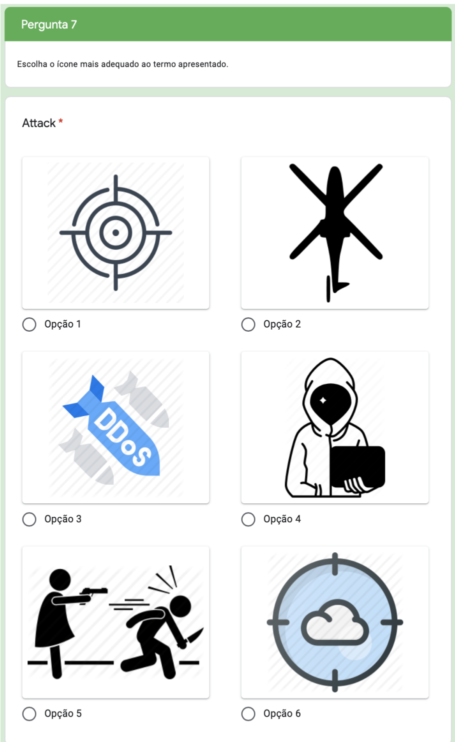
Figure B.1: Complete survey regarding the Term-Icon Connection.



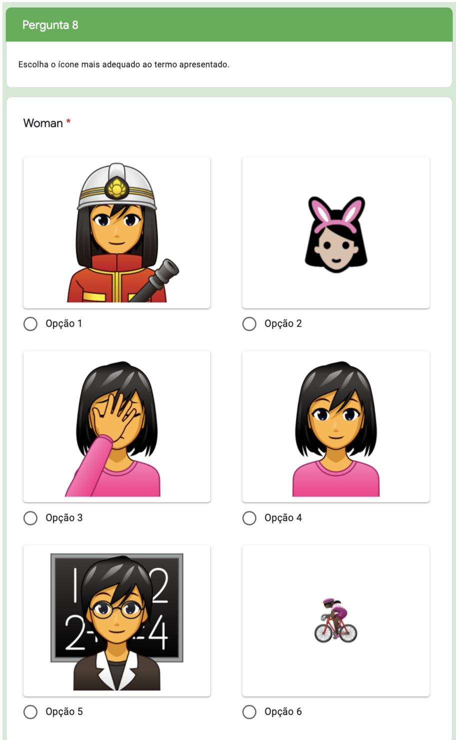
(g) Question 5.



(h) Question 6.

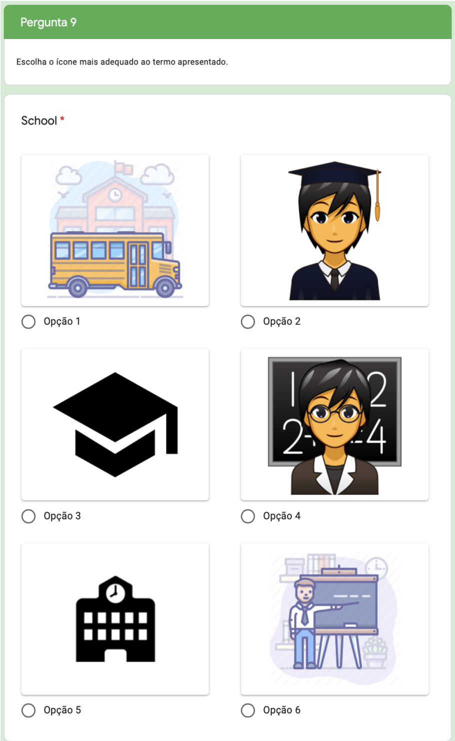


(i) Question 7.

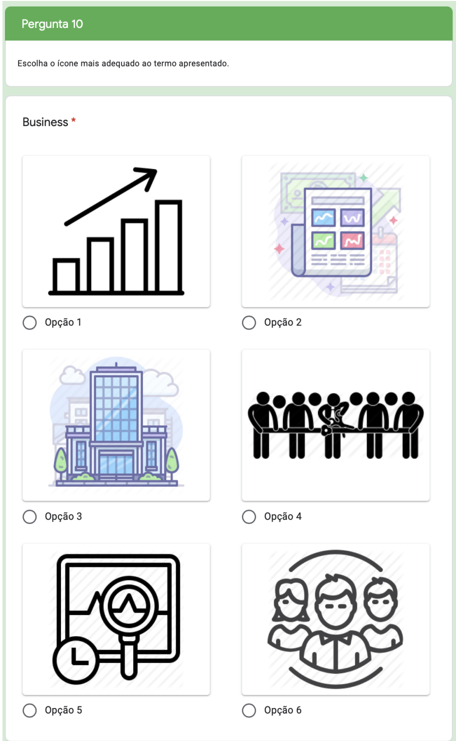


(j) Question 8.

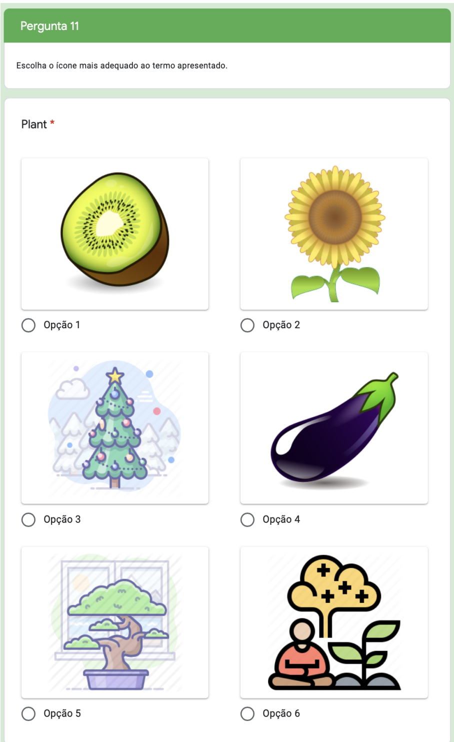
Figure B.1: Complete survey regarding the Term-Icon Connection.



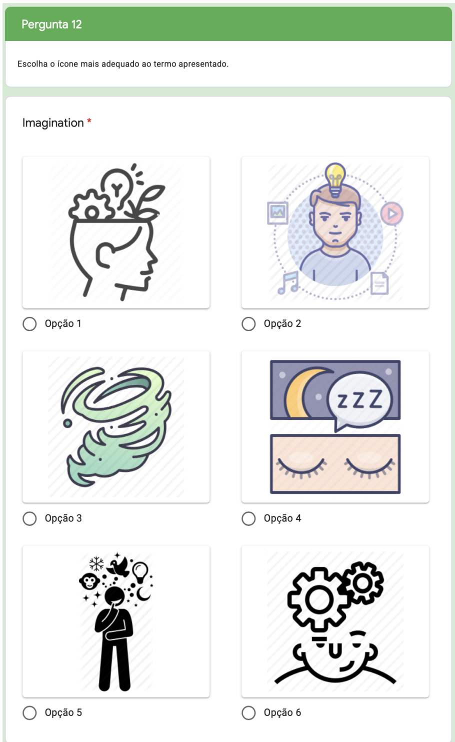
(k) Question 9.



(l) Question 10.

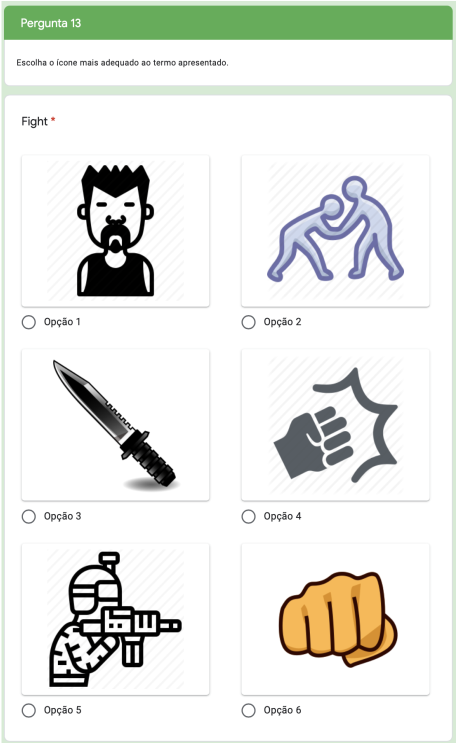


(m) Question 11.

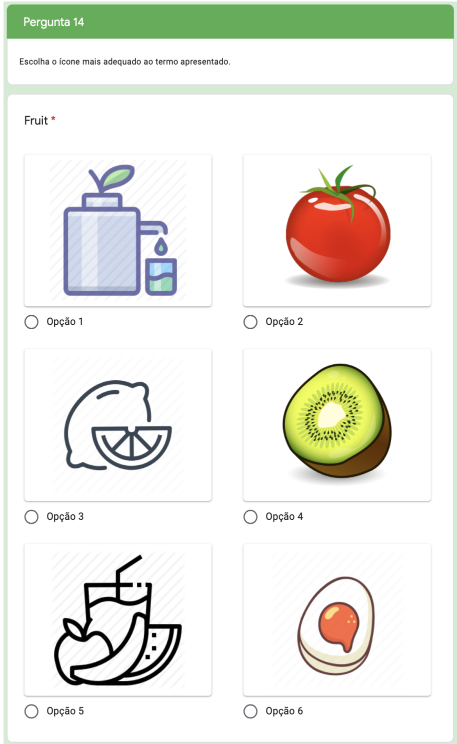


(n) Question 12.

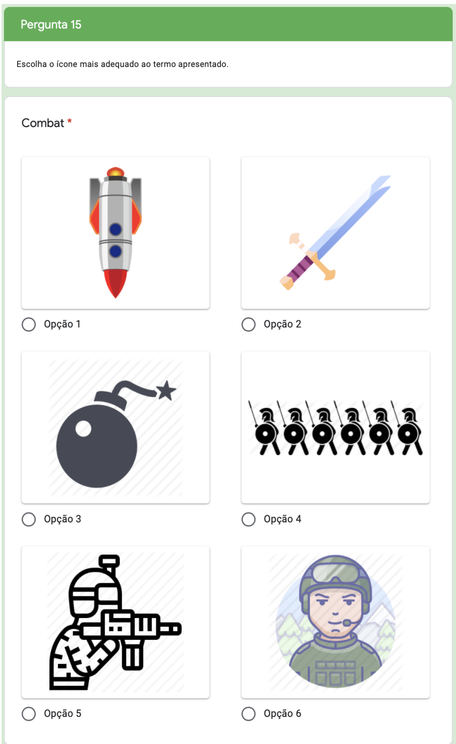
Figure B.1: Complete survey regarding the Term-Icon Connection.



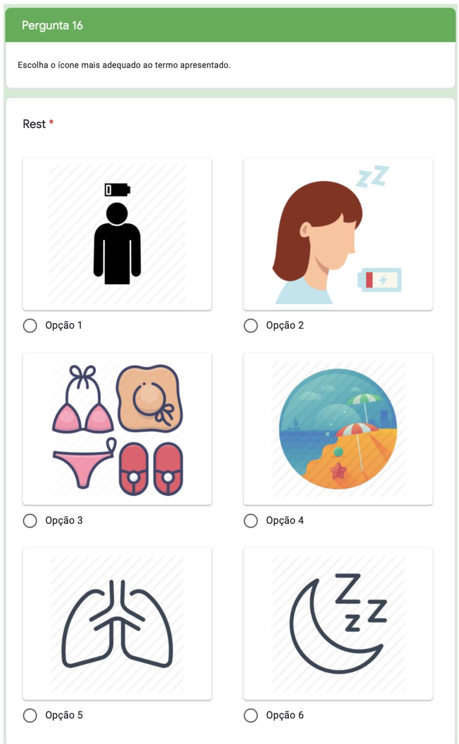
(o) Question 13.



(p) Question 14.

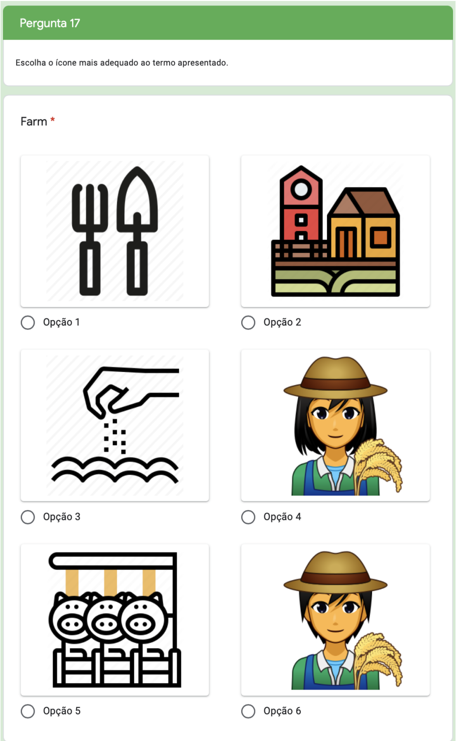


(q) Question 15.

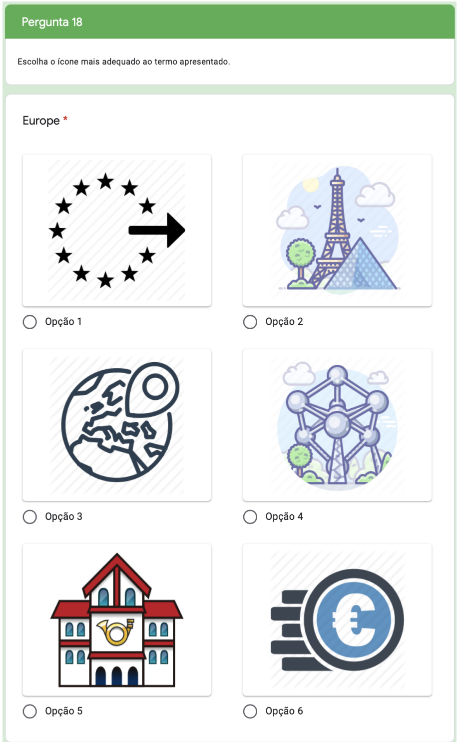


(r) Question 16.

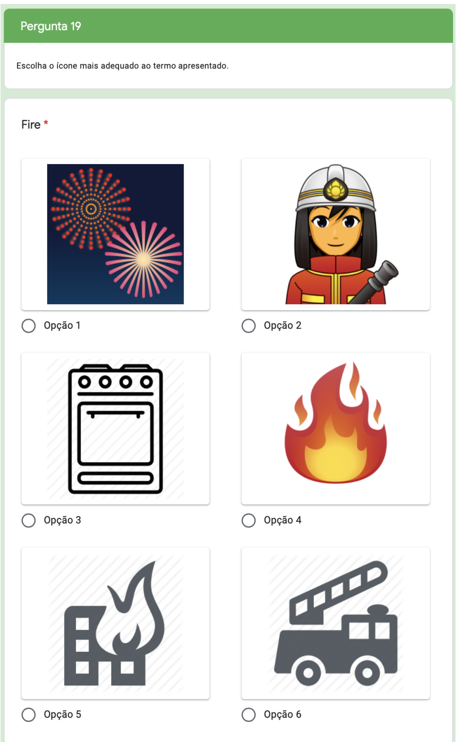
Figure B.1: Complete survey regarding the Term-Icon Connection.



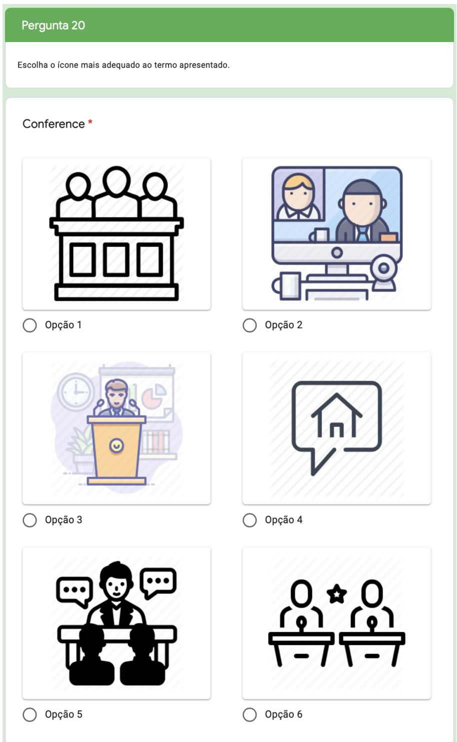
(s) Question 17.



(t) Question 18.



(u) Question 19.



(v) Question 20.

Figure B.1: Complete survey regarding the Term-Icon Connection.

Bibliography

[1] Eneko Agirre, Lluís Màrquez, and Richard Wicentowski, editors. *Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007)*, Prague, Czech Republic, June 2007. Association for Computational Linguistics.

[2] James Allan, Jaime Carbonell, George Doddington, Jonathan Yamron, and Yiming Yang. Topic detection and tracking pilot study final report. *Proceedings of the DARPA Broadcast News Transcription and Understanding Workshop*, February 1998.

[3] José Alves and Célia Rodrigues. Storytelling through infographics. In Ricardo Campos, Alípio Mário Jorge, Adam Jatowt, Sumit Bhatia, and Mark A. Finlayson, editors, *Proceedings of Text2Story - Fourth Workshop on Narrative Extraction From Texts held in conjunction with the 43rd European Conference on Information Retrieval (ECIR 2021), Lucca, Italy, April 1, 2021 (online event due to Covid-19 outbreak)*, volume 2860 of *CEUR Workshop Proceedings*, pages 1–2. CEUR-WS.org, 2021.

[4] Evelin Amorim, Alexandre Ribeiro, Brenda Salenave Santana, Inês Cantante, Alípio Jorge, Sérgio Nunes, Purificação Silvano, Antonio Leal, and Ricardo Campos. Brat2viz: a tool and pipeline for visualizing narratives from annotated texts. In Ricardo Campos, Alípio Mário Jorge, Adam Jatowt, Sumit Bhatia, and Mark A. Finlayson, editors, *Proceedings of Text2Story - Fourth Workshop on Narrative Extraction From Texts held in conjunction with the 43rd European Conference on Information Retrieval (ECIR 2021), Lucca, Italy, April 1, 2021 (online event due to Covid-19 outbreak)*, volume 2860 of *CEUR Workshop Proceedings*, pages 49–56. CEUR-WS.org, 2021.

[5] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *J. Mach. Learn. Res.*, 3:1137–1155, 2003.

[6] Yoshua Bengio, Aaron C. Courville, and Pascal Vincent. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.*, 35(8):1798–1828, 2013. doi:10.1109/TPAMI.2013.50.

[7] Steven Bethard, Guergana Savova, Martha Palmer, and James Pustejovsky. SemEval-2017 task 12: Clinical TempEval. In *Proceedings of the 11th International Workshop on Semantic Evaluation (SemEval-2017)*, pages 565–572, Vancouver, Canada, August 2017. Association for Computational Linguistics. doi:10.18653/v1/S17-2093.

[8] Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomás Mikolov. Enriching word vectors with subword information. *Trans. Assoc. Comput. Linguistics*, 5:135–146, 2017.

[9] E. Cambria and B. White. Jumping nlp curves: A review of natural language processing research [review article]. *IEEE Computational Intelligence Magazine*, 9(2):48–57, 2014. doi:10.1109/MCI.2014.2307227.

[10] Ricardo Campos, Gaël Dias, Alípio Mário Jorge, and Adam Jatowt. Survey of temporal information retrieval and related applications. *ACM Comput. Surv.*, 47(2):15:1–15:41, 2014. doi:10.1145/2619088.

[11] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. Yake! collection-independent automatic keyword extractor. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval*, pages 806–810, Cham, 2018. Springer International Publishing. ISBN: 978-3-319-76941-7.

[12] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Mário Jorge, Célia Nunes, and Adam Jatowt. A text feature based automatic keyword extraction method for single documents. In Gabriella Pasi, Benjamin Piwowarski, Leif Azzopardi, and Allan Hanbury, editors, *Advances in Information Retrieval - 40th European Conference on IR Research, ECIR 2018, Grenoble, France, March 26-29, 2018, Proceedings*, volume 10772 of *Lecture Notes in Computer Science*, pages 684–691. Springer, 2018. doi:10.1007/978-3-319-76941-7_63.

[13] Ricardo Campos, Vítor Mangaravite, Arian Pasquali, Alípio Jorge, Célia Nunes, and Adam Jatowt. Yake! keyword extraction from single documents using multiple local features. *Information Sciences*, 509:257 – 289, 2020. ISSN: 0020-0255. doi:https://doi.org/10.1016/j.ins.2019.09.013.

[14] Ricardo Campos, Jorge Duque, Tiago Cândido, Jorge Mendes, Gaël Dias, Alípio Jorge, and Célia Nunes. Time-matters: Temporal unfolding of texts. In Djoerd Hiemstra, Marie-Francine Moens, Josiane Mothe, Raffaele Perego, Martin Potthast, and Fabrizio Sebastiani, editors, *Advances in Information Retrieval - 43rd European Conference on IR Research, ECIR 2021, Virtual Event, March 28 - April 1, 2021, Proceedings, Part II*, volume 12657 of *Lecture Notes in Computer Science*, pages 492–497. Springer, 2021. doi:10.1007/978-3-030-72240-1_53.

[15] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2004 shared task: Semantic role labeling. In Hwee Tou Ng and Ellen Riloff, editors, *Proceedings of the Eighth Conference on Computational Natural Language Learning, CoNLL 2004, Held in cooperation with HLT-NAACL 2004, Boston, Massachusetts, USA, May 6-7, 2004*, pages 89–97. ACL, 2004.

[16] Xavier Carreras and Lluís Màrquez. Introduction to the conll-2005 shared task: Semantic role labeling. In Ido Dagan and Daniel Gildea, editors, *Proceedings of the Ninth Conference*

on Computational Natural Language Learning, CoNLL 2005, Ann Arbor, Michigan, USA, June 29-30, 2005, pages 152–164. ACL, 2005.

[17] David Caswell and Konstantin Dörr. Automated journalism 2.0: Event-driven narratives. *Journalism Practice*, 12(4):477–496, May 2017. doi:10.1080/17512786.2017.1320773.

[18] Angel X. Chang and Christopher D. Manning. Sutime: A library for recognizing and normalizing time expressions. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 3735–3740. European Language Resources Association (ELRA), 2012.

[19] Seymour Chatman. Story and discourse: Narrative structure in fiction and film. *Cornell University Press*, 1980.

[20] Yubo Chen, Liheng Xu, Kang Liu, Daojian Zeng, and Jun Zhao. Event extraction via dynamic multi-pooling convolutional neural networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 167–176. The Association for Computer Linguistics, 2015. doi:10.3115/v1/p15-1017.

[21] Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *CoRR*, abs/1412.3555, 2014.

[22] Jacob Cohen. A coefficient of agreement for nominal scales. volume 20, page 37–46. Educational and Psychological Measurement, 1960.

[23] Jacob Cohen. Weighted kappa: Nominal scale agreement provision for scaled disagreement or partial credit. volume 70, page 213–220. Psychological Bulletin, 1968.

[24] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *CoRR*, abs/1103.0398, 2011.

[25] Linguistic Data Consortium. *ACE (Automatic Content Extraction) English Annotation Guidelines for Events*, 2005. https://www.ldc.upenn.edu/sites/www.ldc.upenn.edu/files/english-events-guidelines-v5.4.3.pdf.

[26] Weiwei Cui, Xiaoyu Zhang, Yun Wang, He Huang, Bei Chen, Lei Fang, Haidong Zhang, Jian-Guan Lou, and Dongmei Zhang. Text-to-viz: Automatic generation of infographics from proportion-related natural language statements. *IEEE Trans. Vis. Comput. Graph.*, 26(1):906–916, 2020. doi:10.1109/TVCG.2019.2934785.

[27] Koen Dercksen and Arjen P. de Vries. First steps towards patient-friendly presentation of dutch radiology reports. In Francisco M. Couto and Martin Krallinger, editors, *Proceedings of the Workshop on Semantic Indexing and Information Retrieval for Health from heterogeneous content types and languages co-located with 42nd European Conference on Information Retrieval, SIIRH@ECIR 2020, Lisbon, Portugal, April 14, 2020*, volume 2619 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2020.

[28] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019. doi:10.18653/v1/n19-1423.

[29] George R. Doddington, Alexis Mitchell, Mark A. Przybocki, Lance A. Ramshaw, Stephanie M. Strassel, and Ralph M. Weischedel. The automatic content extraction (ACE) program - tasks, data, and evaluation. In *Proceedings of the Fourth International Conference on Language Resources and Evaluation, LREC 2004, May 26-28, 2004, Lisbon, Portugal*. European Language Resources Association, 2004.

[30] Tamas E. Doszkocs. Natural language processing in information retrieval. *Journal of the American Society for Information Science*, 37(4):191–196, July 1986. doi:10.1002/(sici)1097-4571(198607)37:4<191::aid-asi3>3.0.co;2-y.

[31] Zi-Yi Dou, Pengfei Liu, Hiroaki Hayashi, Zhengbao Jiang, and Graham Neubig. Gsum: A general framework for guided neural abstractive summarization. *CoRR*, abs/2010.08014, 2020.

[32] Aleksei Dudchenko, Matthias Ganzinger, and Georgy Kopanitsa. Diagnoses detection in short snippets of narrative medical texts. *Procedia Computer Science*, 156:150–157, 2019. doi:10.1016/j.procs.2019.08.190.

[33] Magali Sanches Duran and Sandra Maria Aluísio. Propbank-br: a brazilian treebank annotated with semantic role labels. In Nicoletta Calzolari (Conference Chair), Khalid Choukri, Thierry Declerck, Mehmet Uğur Doğan, Bente Maegaard, Joseph Mariani, Asuncion Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)*, Istanbul, Turkey, may 2012. European Language Resources Association (ELRA). ISBN: 978-2-9517408-7-7.

[34] Lisa Ehrlinger and Wolfram Wöß. Towards a definition of knowledge graphs. In *SEMANTiCS(Posters, Demos, SuCCESS)*, September 2016.

[35] Mahmoud El-Haj, Vasiliki Athanasakou, Sira Ferradans, Catherine Salzedo, Ans Elhag, Houda Bouamor, Marina Litvak, Paul Rayson, George Giannakopoulos, and Nikiforos

Pittaras, editors. *Proceedings of the 1st Joint Workshop on Financial Narrative Processing and MultiLing Financial Summarisation*, Barcelona, Spain (Online), December 2020. COLING.

[36] Dumitru Erhan, Yoshua Bengio, Aaron C. Courville, Pierre-Antoine Manzagol, Pascal Vincent, and Samy Bengio. Why does unsupervised pre-training help deep learning? *J. Mach. Learn. Res.*, 11:625–660, 2010.

[37] Xiaocheng Feng, Lifu Huang, Duyu Tang, Heng Ji, Bing Qin, and Ting Liu. A language-independent neural network for event detection. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 2: Short Papers*. The Association for Computer Linguistics, 2016. doi:10.18653/v1/p16-2011.

[38] Xiaocheng Feng, Bing Qin, and Ting Liu. A language-independent neural network for event detection. *Sci. China Inf. Sci.*, 61(9):092106:1–092106:12, 2018. doi:10.1007/s11432-017-9359-x.

[39] Ana Figueiras. Narrative visualization: A case study of how to incorporate narrative elements in existing visualizations. In Ebad Banissi, Mark W. McK. Bannatyne, Francis T. Marchese, Muhammad Sarfraz, Anna Ursyn, Gilles Venturini, Theodor G. Wyeld, Urska Cvek, Marjan Trutschl, Georges G. Grinstein, Vladimir Geroimenko, Sarah Kenderdine, and Fatma Bouali, editors, *18th International Conference on Information Visualisation, IV 2014, Paris, France, July 16-18, 2014*, pages 46–52. IEEE Computer Society, 2014. doi:10.1109/IV.2014.79.

[40] J. L. Fleiss. Measuring nominal scale agreement among many raters. volume 76, page 378–382. Psychological Bulletin, 1971.

[41] E. R. Fonseca and J. L. G. Rosa. A two-step convolutional neural network approach for semantic role labeling. In *The 2013 International Joint Conference on Neural Networks (IJCNN)*, pages 1–7, 2013. doi:10.1109/IJCNN.2013.6707118.

[42] International Organization for Standardization. ISO/WD 24617-1, language resource management—semantic annotation framework (semaf)—part 1: Time and events. 2007.

[43] International Organization for Standardization. ISO 24617-9, language resource management— semantic annotation framework —part 9: Reference annotation framework (RAF). 2019.

[44] Michael Färber, Basil Ell, Carsten Menne, Achim Rettinger, and Frederic Bartschererr. Linked data quality of dbpedia, freebase, opencyc, wikidata, and yago. *Semantic Web Journal*, 2016.

[45] Bart Geurts, David I. Beaver, and Emar Maier. Discourse Representation Theory. In Edward N. Zalta, editor, *The Stanford Encyclopedia of Philosophy*. Metaphysics Research Lab, Stanford University, spring 2020 edition, 2020.

[46] Yoav Goldberg. A primer on neural network models for natural language processing. *J. Artif. Intell. Res.*, 57:345–420, 2016. doi:10.1613/jair.4992.

[47] Daniel Gomes, David Cruz, João Miranda, Miguel Costa, and Simão Fontes. Search the past with the portuguese web archive. In Leslie Carr, Alberto H. F. Laender, Bernadette Farias Lóscio, Irwin King, Marcus Fontoura, Denny Vrandecic, Lora Aroyo, José Palazzo M. de Oliveira, Fernanda Lima, and Erik Wilde, editors, *22nd International World Wide Web Conference, WWW '13, Rio de Janeiro, Brazil, May 13-17, 2013, Companion Volume*, pages 321–324. International World Wide Web Conferences Steering Committee / ACM, 2013. doi:10.1145/2487788.2487934.

[48] Ralph Grishman and Beth Sundheim. Message understanding conference- 6: A brief history. In *16th International Conference on Computational Linguistics, Proceedings of the Conference, COLING 1996, Center for Sprogteknologi, Copenhagen, Denmark, August 5-9, 1996*, pages 466–471, 1996.

[49] Max Grusky, Mor Naaman, and Yoav Artzi. Newsroom: A dataset of 1.3 million summaries with diverse extractive strategies. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 708–719. Association for Computational Linguistics, 2018. doi:10.18653/v1/n18-1065.

[50] Jan Hajic, Massimiliano Ciaramita, Richard Johansson, Daisuke Kawahara, Maria Antònia Martí, Lluís Màrquez, Adam Meyers, Joakim Nivre, Sebastian Padó, Jan Stepánek, Pavel Stranák, Mihai Surdeanu, Nianwen Xue, and Yi Zhang. The conll-2009 shared task: Syntactic and semantic dependencies in multiple languages. In Jan Hajic, editor, *Proceedings of the Thirteenth Conference on Computational Natural Language Learning: Shared Task, CoNLL 2009, Boulder, Colorado, USA, June 4, 2009*, pages 1–18. ACL, 2009.

[51] David Harel and P. S. Thiagarajan. *Message Sequence Charts*, pages 77–105. Springer US, Boston, MA, 2003. ISBN: 978-0-306-48738-5. doi:10.1007/0-306-48738-1_4.

[52] Lane Harrison, Katharina Reinecke, and Remco Chang. Infographic aesthetics: Designing for the first impression. In Bo Begole, Jinwoo Kim, Kori Inkpen, and Woontack Woo, editors, *Proceedings of the 33rd Annual ACM Conference on Human Factors in Computing Systems, CHI 2015, Seoul, Republic of Korea, April 18-23, 2015*, pages 1187–1190. ACM, 2015. doi:10.1145/2702123.2702545.

[53] Dan Hendrycks and Thomas G. Dietterich. Benchmarking neural network robustness to common corruptions and surface variations. *ICLR 2019*, 2019.

[54] Swapnil Hingmire, Nitin Ramrakhiyani, Avinash Kumar Singh, Sangameshwar Patil, Girish Keshav Palshikar, Pushpak Bhattacharyya, and Vasudeva Varma. Extracting message sequence charts from hindi narrative text. In Claire Bonial, Tommaso Caselli, Snigdha Chaturvedi, Elizabeth Clark, Ruihong Huang, Mohit Iyyer, Alejandro Jaimes,

Heng Ji, Lara J. Martin, Ben Miller, Teruko Mitamura, Nanyun Peng, and Joel R. Tetreault, editors, *Proceedings of the First Joint Workshop on Narrative Understanding, Storylines, and Events, NUSE@ACL 2020, Online, July 9, 2020*, pages 87–96. Association for Computational Linguistics, 2020. doi:10.18653/v1/2020.nuse-1.11.

[55] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural Comput.*, 9 (8):1735–1780, 1997. doi:10.1162/neco.1997.9.8.1735.

[56] Kun Jing and Jungang Xu. A survey on neural network language models. *CoRR*, abs/1906.03591, 2019.

[57] Dan Jurafsky and James H. Martin. *Speech and language processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd Edition draft*, chapter 2, pages 2–28. Prentice Hall, December 2020. ISBN: 978-0-13-095069-7.

[58] Dan Jurafsky and James H. Martin. *Speech and language processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd Edition draft*, chapter 3, pages 29–55. Prentice Hall, December 2020. ISBN: 978-0-13-095069-7.

[59] Dan Jurafsky and James H. Martin. *Speech and language processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd Edition draft*, chapter 6, pages 96–126. Prentice Hall, December 2020. ISBN: 978-0-13-095069-7.

[60] Dan Jurafsky and James H. Martin. *Speech and language processing - An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition, 3rd Edition draft*, chapter 19, pages 373–392. Prentice Hall, December 2020. ISBN: 978-0-13-095069-7.

[61] Nattiya Kanhabua, Roi Blanco, and Kjetil Nørvåg. Temporal information retrieval. *Found. Trends Inf. Retr.*, 9(2):91–208, 2015. doi:10.1561/1500000043.

[62] Tanja Keller and Sigmar-Olaf Tergan. Visualizing knowledge and information: An introduction. In Sigmar-Olaf Tergan and Tanja Keller, editors, *Knowledge and Information Visualization, Searching for Synergies [outcome of a workshop held in Tübingen, Germany, May 2004]*, volume 3426 of *Lecture Notes in Computer Science*, pages 1–23. Springer, 2005. doi:10.1007/11510154_1.

[63] Yoon Kim. Convolutional neural networks for sentence classification. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1746–1751. ACL, 2014. doi:10.3115/v1/d14-1181.

[64] Yoon Kim, Yacine Jernite, David A. Sontag, and Alexander M. Rush. Character-aware neural language models. In Dale Schuurmans and Michael P. Wellman, editors, *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence, February 12-17, 2016, Phoenix, Arizona, USA*, pages 2741–2749. AAAI Press, 2016.

[65] Thomas N. Kipf and Max Welling. Semi-supervised classification with graph convolutional networks. In *5th International Conference on Learning Representations, ICLR 2017, Toulon, France, April 24-26, 2017, Conference Track Proceedings*. OpenReview.net, 2017.

[66] J. Richard Landis and Gary G. Koch. The measurement of observer agreement for categorical data. volume 33, pages 159–174. Biometrics, 1977.

[67] Julia Langer and Michael Zeiller. Evaluation of the user experience of interactive infographics in online newspapers. In Wolfgang Aigner, Thomas Moser, Kerstin Blumenstein, Matthias Zeppelzauer, Michael Iber, and Grischa Schmiedl, editors, *Proceedings of the 10th Forum Media Technology and 3rd All Around Audio Symposium, St. Pölten, Austria, November 29-30, 2017*, volume 2009 of *CEUR Workshop Proceedings*, pages 97–106. CEUR-WS.org, 2017.

[68] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. *CoRR*, abs/1910.13461, 2019.

[69] Hui Li, Jannik Strötgen, Julian Zell, and Michael Gertz. Chinese temporal tagging with HeidelTime. In *Proceedings of the 14th Conference of the European Chapter of the Association for Computational Linguistics, volume 2: Short Papers*, pages 133–137, Gothenburg, Sweden, April 2014. Association for Computational Linguistics. doi:10.3115/v1/E14-4026.

[70] Pinjie Lv, Xinyue Wang, and Chengqi Xue. Research on automatic recognition method of icon style. In *22nd IEEE International Conference on Industrial Technology, ICIT 2021, Valencia, Spain, March 10-12, 2021*, pages 931–935. IEEE, 2021. doi:10.1109/ICIT46573.2021.9453509.

[71] Inderjeet Mani and D. George Wilson. Robust temporal processing of news. In *38th Annual Meeting of the Association for Computational Linguistics, Hong Kong, China, October 1-8, 2000*, pages 69–76. ACL, 2000. doi:10.3115/1075218.1075228.

[72] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to information retrieval*. Cambridge University Press, 2008. ISBN: 978-0-521-86571-5. doi:10.1017/CBO9780511809071.

[73] Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. Building a large annotated corpus of english: The penn treebank. *Comput. Linguistics*, 19(2):313–330, 1993.

[74] Lluís Màrquez, Xavier Carreras, Kenneth C. Litkowski, and Suzanne Stevenson. Semantic role labeling: An introduction to the special issue. *Comput. Linguistics*, 34(2):145–159, 2008. doi:10.1162/coli.2008.34.2.145.

[75] Bryan McCann, James Bradbury, Caiming Xiong, and Richard Socher. Learned in translation: Contextualized word vectors. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 6294–6305, 2017.

[76] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, pages 3111–3119, 2013.

[77] Tomas Mikolov, Edouard Grave, Piotr Bojanowski, Christian Puhrsch, and Armand Joulin. Advances in pre-training distributed word representations. In *Proceedings of the International Conference on Language Resources and Evaluation (LREC 2018)*, 2018.

[78] Derek Miller. Leveraging BERT for extractive text summarization on lectures. *CoRR*, abs/1906.04165, 2019.

[79] Ramesh Nallapati, Feifei Zhai, and Bowen Zhou. Summarunner: A recurrent neural network based sequence model for extractive summarization of documents. *CoRR*, abs/1611.04230, 2016.

[80] Usman Naseem, Imran Razzak, Shah Khalid Khan, and Mukesh Prasad. A comprehensive survey on word representation models: From classical to state-of-the-art word representation language models. *CoRR*, abs/2010.15036, 2020.

[81] Girish Palshikar, Sachin Pawar, Sangameshwar Patil, Swapnil Hingmire, Nitin Ramrakhiyani, Harsimran Bedi, Pushpak Bhattacharyya, and Vasudeva Varma. Extraction of message sequence charts from narrative history text. In *Proceedings of the First Workshop on Narrative Understanding*, pages 28–36, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. doi:10.18653/v1/W19-2404.

[82] Arian Pasquali, Vítor Mangaravite, Ricardo Campos, Alípio Mário Jorge, and Adam Jatowt. Interactive system for automatically generating temporal narratives. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part II*, volume 11438 of *Lecture Notes in Computer Science*, pages 251–255. Springer, 2019. doi:10.1007/978-3-030-15719-7_34.

[83] Heiko Paulheim. Knowledge graph refinement: A survey of approaches and evaluation methods. *Semantic Web*, 8(3):489–508, 2017. doi:10.3233/SW-160218.

[84] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In Alessandro Moschitti, Bo Pang, and Walter Daelemans, editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing, EMNLP 2014, October 25-29, 2014, Doha, Qatar, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1532–1543. ACL, 2014. doi:10.3115/v1/d14-1162.

[85] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 2227–2237. Association for Computational Linguistics, 2018. doi:10.18653/v1/n18-1202.

[86] Volha Petukhova and Harry Bunt. LIRICS semantic role annotation: Design and evaluation of a set of data categories. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco*. European Language Resources Association, 2008.

[87] Jakub Piskorski, Vanni Zavarella, Martin Atkinson, and Marco Verile. Timelines: Entity-centric event extraction from online news. In Ricardo Campos, Alípio Mário Jorge, Adam Jatowt, and Sumit Bhatia, editors, *Proceedings of Text2Story - Third Workshop on Narrative Extraction From Texts co-located with 42nd European Conference on Information Retrieval, Text2Story@ECIR 2020, Lisbon, Portugal, April 14th, 2020 [online only]*, volume 2593 of *CEUR Workshop Proceedings*, pages 105–114. CEUR-WS.org, 2020.

[88] Jakub Piskorski, Vanni Zavarella, Martin Atkinson, and Marco Verile. Timelines: Entity-centric event extraction from online news. In Ricardo Campos, Alípio Mário Jorge, Adam Jatowt, and Sumit Bhatia, editors, *Proceedings of Text2Story - Third Workshop on Narrative Extraction From Texts co-located with 42nd European Conference on Information Retrieval, Text2Story@ECIR 2020, Lisbon, Portugal, April 14th, 2020 [online only]*, volume 2593 of *CEUR Workshop Proceedings*, pages 105–114. CEUR-WS.org, 2020.

[89] Sameer Pradhan, Alessandro Moschitti, Nianwen Xue, Hwee Tou Ng, Anders Björkelund, Olga Uryupina, Yuchen Zhang, and Zhi Zhong. Towards robust linguistic analysis using ontonotes. In Julia Hockenmaier and Sebastian Riedel, editors, *Proceedings of the Seventeenth Conference on Computational Natural Language Learning, CoNLL 2013, Sofia, Bulgaria, August 8-9, 2013*, pages 143–152. ACL, 2013.

[90] Gerald Prince. A dictionary of narratology. *University of Nebraska Press, Lincoln*, 1987.

[91] Jay Pujara, Hui Miao, Lise Getoor, and William W. Cohen. Knowledge graph identification. In Harith Alani, Lalana Kagal, Achille Fokoue, Paul Groth, Chris Biemann, Josiane Xavier

Parreira, Lora Aroyo, Natasha F. Noy, Chris Welty, and Krzysztof Janowicz, editors, *The Semantic Web - ISWC 2013 - 12th International Semantic Web Conference, Sydney, NSW, Australia, October 21-25, 2013, Proceedings, Part I*, volume 8218 of *Lecture Notes in Computer Science*, pages 542–557. Springer, 2013. doi:10.1007/978-3-642-41335-3_34.

[92] James Pustejovsky and Nikhil Krishnaswamy. VoxML: A visualization modeling language. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pages 4606–4613, Portorož, Slovenia, May 2016. European Language Resources Association (ELRA).

[93] James Pustejovsky, Robert Knippen, Jessica Littman, and Roser Saurí. Temporal and event information in natural language text. *Lang. Resour. Evaluation*, 39(2-3):123–164, 2005. doi:10.1007/s10579-005-7882-7.

[94] James Pustejovsky, Marc Verhagen, Roser Sauri, Jessica Littman, Gaizauskas Robert, Katz Graham, Inderjeet Mani, Robert Knippen, and Andrea Setzer. Timebank 1.2, 2006. doi:10.35111/09B1-5N19.

[95] Xipeng Qiu, Tianxiang Sun, Yige Xu, Yunfan Shao, Ning Dai, and Xuanjing Huang. Pre-trained models for natural language processing: A survey. *CoRR*, abs/2003.08271, 2020.

[96] Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. Improving language understanding by generative pre-training. In *OpenAI 1–10*, 2018.

[97] Aditya Ramesh, Mikhail Pavlov, Gabriel Goh, Scott Gray, Chelsea Voss, Alec Radford, Mark Chen, and Ilya Sutskever. Zero-shot text-to-image generation. In Marina Meila and Tong Zhang, editors, *Proceedings of the 38th International Conference on Machine Learning*, volume 139 of *Proceedings of Machine Learning Research*, pages 8821–8831. PMLR, 18–24 Jul 2021.

[98] Jens Helge Reelfs, Oliver Hohlfeld, Markus Strohmaier, and Niklas Henckell. Word-emoji embeddings from large scale messaging data reflect real-world semantic associations of expressive icons. *CoRR*, abs/2006.01207, 2020.

[99] M. O. Riedl and R. M. Young. Narrative planning: Balancing plot and character. *Journal of Artificial Intelligence Research*, 39:217–268, Sep 2010. ISSN: 1076-9757. doi:10.1613/jair.2989.

[100] Stuart Rose, Dave Engel, Nick Cramer, and Wendy Cowley. Automatic keyword extraction from individual documents. In *Text Mining*, pages 1–20. John Wiley & Sons, Ltd, March 2010. doi:10.1002/9780470689646.ch1.

[101] Karin K. Schuler. *VerbNet: A broad-coverage, comprehensive verb lexicon*. PhD thesis, 2005.

[102] David Semedo and João Magalhães. Dynamic-keyword extraction from social media. In Leif Azzopardi, Benno Stein, Norbert Fuhr, Philipp Mayr, Claudia Hauff, and Djoerd Hiemstra, editors, *Advances in Information Retrieval - 41st European Conference on IR Research, ECIR 2019, Cologne, Germany, April 14-18, 2019, Proceedings, Part I*, volume 11437 of *Lecture Notes in Computer Science*, pages 852–860. Springer, 2019. doi:10.1007/978-3-030-15712-8_62.

[103] Rico Sennrich, Barry Haddow, and Alexandra Birch. Neural machine translation of rare words with subword units. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*. The Association for Computer Linguistics, 2016. doi:10.18653/v1/p16-1162.

[104] Amrollah Shamsi, Donna Frederick, and Reza BasirianJahromi. Emoji as an aid for information retrieval: A viewpoint regarding metadata, technology and improving international information seeking. *Journal of Library Metadata*, 0(0):1–6, 2021. doi:10.1080/19386389.2020.1907893.

[105] Prafull Sharma and Yingbo Li. Self-supervised contextual keyword and keyphrase retrieval with self-labelling. *Preprints*, August 2019.

[106] Peng Shi and Jimmy Lin. Simple BERT models for relation extraction and semantic role labeling. *CoRR*, 2019.

[107] Purificação Silvano, Antonio Leal, Fátima Silva, Inês Cantante, Fátima Oliveira, and Alípio Jorge. Developing a multilayer semantic annotation scheme based on iso standards for the visualization of a newswire corpus. In *Proceedings of the 17th Joint ACL - ISO Workshop on Interoperable Semantic Annotation*, Association for Computational Linguistics, page 1–13, June 2021.

[108] Amit Singhal. Introducing the knowledge graph: things, not strings. May 2012. (Online; accessed 25 June, 2021). https://blog.google/products/search/introducing-knowledge-graph-things-not/.

[109] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D. Manning, Andrew Y. Ng, and Christopher Potts. Recursive deep models for semantic compositionality over a sentiment treebank. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1631–1642. ACL, 2013.

[110] A. Sofia Medeiros Oliveira. Semantic role labeling in portuguese: Improving the state of the art with transfer learning and bert-based models. Master's thesis, 2020.

[111] Devin Soni. *Supervised vs. Unsupervised Learning*, Mar 2018. (Online; accessed 1 July, 2021). https://towardsdatascience.com/supervised-vs-unsupervised-learning-14f68e32ea8d.

[112] Pontus Stenetorp, Sampo Pyysalo, Goran Topić, Tomoko Ohta, Sophia Ananiadou, and Jun'ichi Tsujii. brat: a web-based tool for NLP-assisted text annotation. In *Proceedings of the Demonstrations at the 13th Conference of the European Chapter of the Association for Computational Linguistics*, pages 102–107, Avignon, France, April 2012. Association for Computational Linguistics.

[113] Jannik Strötgen and Michael Gertz. Heideltime: High quality rule-based extraction and normalization of temporal expressions. In Katrin Erk and Carlo Strapparava, editors, *Proceedings of the 5th International Workshop on Semantic Evaluation, SemEval@ACL 2010, Uppsala University, Uppsala, Sweden, July 15-16, 2010*, pages 321–324. The Association for Computer Linguistics, 2010.

[114] Jannik Strötgen and Michael Gertz. Temporal tagging on different domains: Challenges, strategies, and gold standards. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Mehmet Ugur Dogan, Bente Maegaard, Joseph Mariani, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Eighth International Conference on Language Resources and Evaluation, LREC 2012, Istanbul, Turkey, May 23-25, 2012*, pages 3746–3753. European Language Resources Association (ELRA), 2012.

[115] Jannik Strötgen and Michael Gertz. Multilingual and cross-domain temporal tagging. *Lang. Resour. Evaluation*, 47(2):269–298, 2013. doi:10.1007/s10579-012-9179-y.

[116] Jannik Strötgen and Michael Gertz. A baseline temporal tagger for all languages. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 541–547, Lisbon, Portugal, September 2015. Association for Computational Linguistics. doi:10.18653/v1/D15-1063.

[117] Jannik Strötgen, Ayser Armiti, Tran Van Canh, Julian Zell, and Michael Gertz. Time for more languages: Temporal tagging of arabic, italian, spanish, and vietnamese. *ACM Trans. Asian Lang. Inf. Process.*, 13(1):1:1–1:21, 2014. doi:10.1145/2540989.

[118] Jannik Strötgen, Thomas Bögel, Julian Zell, Ayser Armiti, Tran Van Canh, and Michael Gertz. Extending heideltime for temporal expressions referring to historic dates. In Nicoletta Calzolari, Khalid Choukri, Thierry Declerck, Hrafn Loftsson, Bente Maegaard, Joseph Mariani, Asunción Moreno, Jan Odijk, and Stelios Piperidis, editors, *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC 2014, Reykjavik, Iceland, May 26-31, 2014*, pages 2390–2397. European Language Resources Association (ELRA), 2014.

[119] Lima Subramanian and R.S Karthik. KEYWORD EXTRACTION: A COMPARATIVE STUDY USING GRAPH BASED MODEL AND RAKE. *International Journal of Advanced Research*, 5(3):1133–1137, March 2017. doi:10.21474/ijar01/3616.

[120] Mihai Surdeanu, Richard Johansson, Adam Meyers, Lluís Màrquez, and Joakim Nivre. The conll 2008 shared task on joint parsing of syntactic and semantic dependencies. In Alexander Clark and Kristina Toutanova, editors, *Proceedings of the Twelfth Conference on*

Computational Natural Language Learning, CoNLL 2008, Manchester, UK, August 16-17, 2008, pages 159–177. ACL, 2008.

[121] Kai Sheng Tai, Richard Socher, and Christopher D. Manning. Improved semantic representations from tree-structured long short-term memory networks. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL 2015, July 26-31, 2015, Beijing, China, Volume 1: Long Papers*, pages 1556–1566. The Association for Computer Linguistics, 2015. doi:10.3115/v1/p15-1150.

[122] Erik F. Tjong Kim Sang and Fien De Meulder. Introduction to the CoNLL-2003 shared task: Language-independent named entity recognition. In *Proceedings of the Seventh Conference on Natural Language Learning at HLT-NAACL 2003*, pages 142–147, 2003.

[123] Peter D. Turney. Learning algorithms for keyphrase extraction. *Inf. Retr.*, 2(4):303–336, 2000. doi:10.1023/A:1009976227802.

[124] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 30: Annual Conference on Neural Information Processing Systems 2017, December 4-9, 2017, Long Beach, CA, USA*, pages 5998–6008, 2017.

[125] Marc Verhagen and James Pustejovsky. Temporal processing with the TARSQI toolkit. In Allan Ramsay and Kalina Bontcheva, editors, *COLING 2008, 22nd International Conference on Computational Linguistics, Demo Proceedings, 18-22 August 2008, Manchester, UK*, pages 189–192, 2008.

[126] Marc Verhagen, Rob Gaizauskas, Frank Schilder, M. Hepple, Jessica Moszkowicz, and James Pustejovsky. The tempeval challenge: Identifying temporal relations in text. *Language Resources and Evaluation*, 43:161–179, 06 2009. doi:10.1007/s10579-009-9086-z.

[127] Marc Verhagen, Roser Saurí, Tommaso Caselli, and James Pustejovsky. SemEval-2010 task 13: TempEval-2. In *Proceedings of the 5th International Workshop on Semantic Evaluation*, pages 57–62, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

[128] Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. KEA: practical automatic keyphrase extraction. *CoRR*, cs.DL/9902007, 1999.

[129] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on*

Empirical Methods in Natural Language Processing: System Demonstrations, pages 38–45, Online, October 2020. Association for Computational Linguistics.

[130] Jiacheng Xu, Zhe Gan, Yu Cheng, and Jingjing Liu. Discourse-aware neural extractive text summarization. In Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, pages 5021–5031. Association for Computational Linguistics, 2020. doi:10.18653/v1/2020.acl-main.451.

[131] Wei Xu and Alex Rudnicky. Can artificial neural networks learn language models? In *Sixth International Conference on Spoken Language Processing, ICSLP 2000 / INTERSPEECH 2000, Beijing, China, October 16-20, 2000*, pages 202–205. ISCA, 2000.

[132] Ming Zhong, Pengfei Liu, Danqing Wang, Xipeng Qiu, and Xuanjing Huang. Searching for effective neural extractive summarization: What works and what's next. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 1049–1058, Florence, Italy, July 2019. Association for Computational Linguistics. doi:10.18653/v1/P19-1100.

[133] Qingyu Zhou, Nan Yang, Furu Wei, Shaohan Huang, Ming Zhou, and Tiejun Zhao. Neural document summarization by jointly learning to score and select sentences. In Iryna Gurevych and Yusuke Miyao, editors, *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics, ACL 2018, Melbourne, Australia, July 15-20, 2018, Volume 1: Long Papers*, pages 654–663. Association for Computational Linguistics, 2018. doi:10.18653/v1/P18-1061.

[134] Xiao-Dan Zhu, Parinaz Sobhani, and Hongyu Guo. Long short-term memory over recursive structures. In Francis R. Bach and David M. Blei, editors, *Proceedings of the 32nd International Conference on Machine Learning, ICML 2015, Lille, France, 6-11 July 2015*, volume 37 of *JMLR Workshop and Conference Proceedings*, pages 1604–1612. JMLR.org, 2015.