DEPARTMENT OF
COMPUTER SCIENCE

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DANIEL JOÃO NUNES CASTANHO
Bachelor in Computer Science

STRUCTURING AND ORGANIZING LARGE SCALE GRAPH TEMPORAL INFORMATION

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
October, 2023

NOVA SCHOOL OF
SCIENCE & TECHNOLOGY

DEPARTMENT OF
COMPUTER SCIENCE

STRUCTURING AND ORGANIZING LARGE SCALE GRAPH TEMPORAL INFORMATION

DANIEL JOÃO NUNES CASTANHO

Bachelor in Computer Science

Adviser: David Semedo
Assistant Professor, NOVA University Lisbon

MASTER IN COMPUTER SCIENCE

NOVA University Lisbon
October, 2023

Structuring and Organizing Large Scale Graph Temporal Information

This is dedicated to all those who dare, strive, and inspire -
your journey fuels the spirit of possibility.

Acknowledgements

This thesis may have been developed over the course of a single year, but I feel that many of the hardships and blessings of the past five years have led to this particular moment. The last delivery of this course, after so many before it. As much as this project is my own, one person cannot create alone - others always influence, directly or indirectly; they provide opinions or affect the feelings of the creator - and I am no exception. This work is of others, too many too enumerate, but here, I will do my best.

To begin with, none of it would have been possible without the guidance of my advisor, Professor David Semedo, whose kindness, patience, and experience helped me achieve this goal and to whom I am deeply thankful. Of course, I could not forget to thank Professor João Magalhães, who challenged me to take on this problem and provided invaluable assistance throughout its development.

I am thankful to NOVA School of Science and Technology and the NOVA LINCS research center, who created a great environment for me to work, as well as to the project VisionBox (CC 04040101) and all its participants. Additionally, I would like to thank **João** and **Ricardo** and all the members of team TWIZ for their help and camaraderie.

The development of this work was only possible because of the support my family gave me. My parents, **Lucília** and **Paulo**, and my brother, **Gui**, who have always stood with me, and have helped with even the smallest of things - it is easy to forget the value of a hot meal in a time of academic crisis.

I owe it to all my friends too (João, Alex, Rafa, Carolina, Rafael, Diogo, Sofia, André, Erika, Sílvia, Li, Luís, LH and many more), who pushed me forward and took my mind away from work whenever we were together, so that I could truly relax. In particular, I'd like to thank **Eric**, my brother-in-arms who accompanied me through these past five years closer than almost anyone else and whose realistic and calming views balanced my anxious perfectionism in so many projects and moments together.

My greatest thanks falls on **Catarina**, which I can hardly express in words. She is my rock, pushing me forward into uncharted seas but always giving me a safe shore. Your hardworking nature and keen mind have inspired me more than you could ever know. Thank you.

"If you only do what you can do, you will never be more than you are now." (Master Shifu)

Abstract

The internet is a large-scale web of interconnected pages, which are ephemeral by nature. Most web pages have a short life span, which leads to a loss of data as they die out. To fight this knowledge gap, web archives were created, which occasionally crawl the internet to store web pages. Because of the sheer size of the Internet as well as their longitudinal nature, web archives are some of the largest agglomerations of human social knowledge.

ArquivoPT is the largest Portuguese web archive, and easily the largest collection of unsorted Portuguese text. Its searching mechanisms focus on full-text search, domain and URL filtering, and time ranges. These methods scale well but fail to capture the implicit semantic connections across pages. Web pages, particularly news pages, are connected through mentioning the same entities, directly or indirectly, or through referring to the same or preceding/succeeding events. These relations between pages form an implicit graph structure that goes unexplored with conventional methods, but that theoretically allows for powerful exploration of knowledge in regards to entities and their relations.

We hypothesize that by making these relations explicit through a knowledge graph, we can perform semantic-based, relation-oriented search and question answering over a news set. By using a natural language processing pipeline to extract entities and relations from Portuguese text, the sources of these connections are made apparent and, in combination with a well-defined model, structured into an RDF graph. This graph is later queried and used with specific algorithms which fulfill specific information-seeking use cases.

We thus contribute with a framework that generalizes to other scenarios, archives, and even domains. It is accompanied by an evaluation methodology and dataset, to assess the quality of these knowledge bases. Because news are not self-contained and lack common facts, we also propose several ways to incorporate these into an entity-relation graph, when it is linked to external knowledge bases.

Keywords: Temporal Graph Structuring, Web Archives, Portuguese Language Processing

Resumo

A internet é uma rede de grande escala, formada por páginas interligadas que são de natureza passageira. A maior parte das páginas web tem uma vida curta, o que resulta numa perda de informação à medida que estas vão desaparecendo. Para combater esta discrepância de informação, foram criados os arquivos web, que ocasionalmente guardam páginas da internet. Devido ao tamanho da internet e à natureza longitudinais deste arquivos, acabam por ser das maiores coleções de conhecimento humano disponíveis.

O ArquivoPT é o maior arquivo web português e facilmente a maior coleção de texto não organizado da língua. Os seus mecanismos de busca focam-se em pesquisa por palavras chave, filtragem por URL e intervalos temporais. Estes métodos escalam bem, mas não conseguem capturar as ligações semânticas que implicitamente existem entre as páginas web. Estas páginas, principalmente as de notícias, estão interligadas por menções das mesmas entidades, diretamente ou indiretamente, e descrições dos mesmos eventos, ou da causalidade/sequência destes. Estas ligações formam um grafo implícito, que não é explorado pelos métodos convencionais de busca, mas que teoricamente permite uma poderosa forma de pesquisa baseada em entidades e as suas relações.

A nossa hipótese é que, ao tornar estas relações explicitas com um grafo de conhecimento, conseguimos fazer buscas semânticas e orientadas a relações, assim como questões, a um conjunto de notícias. Através do processamento sequencial de texto em português, entidades e relações são extraídas, e estas ligações são tornadas obvias e combinadas com um modelo bem definido, que as estrutura num grafo RDF. Este grafo é usado para busca e consultado por algoritmos que cumprem certos usos de casos de pesquisa.

Contribuímos assim com um processo capaz de se generalizar para outros cenários, arquivos e até domínios. É acompanhado por uma metodologia de avaliação, e correspondente dataset, que avaliam a qualidade destas bases de conhecimento. Como noticias omitem conhecimento comum, apresentamos também varias formas de incorporar conhecimento externo em grafos entidade-relação ligados a outras bases de conhecimento.

Palavras-chave: Estruturação de Grafos Temporais, Arquivos Web, Processamento da Língua Portuguesa.

Contents

List of Figures

List of Tables

List of Listings

1

Introduction

The web is a virtual space generated by connections between millions of web pages. This space is fluid due to the ephemeral nature of web pages: every minute pages appear and disappear.

Because web pages die at a moment's notice, information that was once available can just as quickly become inaccessible. As a way to preserve it, web archives such as ArquivoPT[1] and the Internet Archive[2] were created, which crawl the internet to store all sorts of web pages - from important sites to degenerate ones. These collections represent some of the largest aggregations of human social knowledge ever created and are used by historians, researchers, and common users alike for various purposes.

Pages in web archives, especially news pages, often refer to the same events, people, companies, and more. This creates implicit connections between pages that mention the same entities and the relations between those entities. Keyword-based searching struggles to capture this semantic richness, which is implicitly shaped like a graph and holds non-trivial relations that span across time and pages.

1.1 Motivation

Preserving web digital information is important to prevent the creation of a knowledge gap between future generations and current events. In 2003 this problem was addressed by UNESCO, who claimed that the loss of any heritage, including digital ones, constitutes an impoverishment of all nations' heritage [70]. Furthermore, UNESCO recognized the importance of archives in the protection of historical information [71].

Many examples of web archives exist, which store millions of files that in turn contain images, videos, links, and text. These collections are massive and longitudinal in nature - the same page is available at different points in time - which makes searching such a large pool difficult.

[1] https://arquivo.pt/
[2] https://archive.org/

ArquivoPT is the biggest Portuguese web archive. It contains files from the Portuguese domain and related pages, which sum up to a *compressed* total of 876 TeraBytes[3]. As such, it ends up being a rich archive of events, capturing how entities (people, countries, etc.) participated in these events and the course of history itself. Due to this, it has been used in social and communication studies, focusing on news across time and the Portuguese language [3, 38].

ArquivoPT's search mechanisms are based on text indexes. These methods are thus limited to searching by their indexed fields, such as time of crawl or full-text search of specific keywords. While scaleable, it is limited, as one must search specific terms or internet domains, even if more complex boolean operators (`not`, `and`, `or`, etc.) are allowed. This creates a barrier when in the discovery phase of investigation, as well as for establishing relations between entities, particularly when they are spread across several articles.

When capturing pages, web archives (including ArquivoPT) store multiple snapshots of the same page across time and crawl through hyper-references to save related pages. Because of this, there is an explicit graph structure already present in the archive - the edges of which connect different snapshots of the same page or relate pages through hyper-references. However, this existing graph structure composed of hyper-references does not capture the semantics of the archived content. Such semantics are implicit and need to be systematically modeled as an explicit graph to be fully taken advantage of.

1.2 Research Hypothesis

The social, political, and economic space changes over time - sometimes rather drastically. These changes are captured by news articles, which describe the affected entities and how their relations with others in that same context are altered. Due to its basis on connections and relations, this data can be shaped as a graph in what is a simplified view of the world. Such a representation is only possible due to the reliable nature of (most) news.

A knowledge graph - a graph in which its data represents *knowledge* - with the goal of representing these spaces, should thus focus on the relations between entities; their strength, evolution over time, and reasoning. Often times news lack common knowledge statements (such as who the current president is) which makes them unsuitable for the "standard" knowledge graphs, composed of static, factual statements. But news are meant for a different purpose altogether - they are meant to give the population an overview of what is happening in the world and capture its changing environments. As such, we chose to model these dynamic changes between relations in the news domain.

A modeled view of the world should be able to respond to queries - either through a structured language, a post-processing algorithm, or a more dynamic visualization - based on entities, relations, and time, such as:

[3] https://sobre.arquivo.pt/pt/imprensa/o-arquivo-pt-em-numeros/

1. How have the relations of Marcelo Rebelo de Sousa with other political parties changed since he was elected?
2. How strongly related are Brazil and India, since 2015?
3. Which news can be attributed to the relationship between the USA and Mexico?
4. When did the relation between Facebook and WhatsApp - that lasts to this day and is rather strong - start?

To properly answer such questions a graph that relates entities is necessary - one that takes time into account in these relations. Implicit data that could not be searched through full-text methods or analyzed without performing heavy research, becomes accessible at the distance of a query. This graph is thus useful for anyone studying or using relations between entities in their works, such as journalists [51].

Knowledge graphs allow the detection of information that would otherwise be difficult to capture such as how two entities that are never mentioned in the same text are related. This is possible through multi-hop queries, which can traverse over the edges of a graph until a connection is found.

Thus, due to the latent semantic structure of web archives and the querying power of graphs, the **research hypothesis supporting this thesis is**:

> *In order to unveil the relational expressiveness of ArquivoPT's news set, which can be explored with semantic-based queries, an entity-relation knowledge graph can be built by extracting information out of archived texts. Structured data is obtained by a natural language processing pipeline that processes Portuguese texts, and is consequently organized into a graph with a single, well-defined schema.*

1.3 Challenges

A work of this dimension was not without its struggles - the challenges described here represent an overview of the difficulties faced during development and are split into three main categories.

Extraction challenges. Extracting information from text - which was itself automatically extracted from websites - is an ongoing research topic. The European branch of the Portuguese has a distinct lack of tools when it comes to performing language processing. The most reliable task in general is named entity extraction, while the rest still present problems of various kinds from low accuracy to slow performance or too much noisy information. These issues limited the tools we could use and the information we could extract, which in turn limited the shape of the graph. To solve this, simpler extraction techniques were used, along with flexible models and processes that could later be adapted to more complex tools.

Size constraint. ArquivoPT's archived data ranges from 1991 until now (2023). This means there is a *vast* amount of information to store and analyze. Thus, this information needed to be selectively added to the system in a well-defined manner. In addition, due to the time span, temporal aggregation of the data was necessary to avoid overloading the knowledge base, which in turn can take different forms that had to be studied.

Modeling issues. News and relations are flexible sources of information, which can be represented and handled in many different ways. Deciding on the specific schema, and way to model and extract relations, took several exploratory attempts and tests before being concluded. This issue extended to the processing pipeline as each different model required a different pipeline to be properly generated. Adhering to an existing model or ontology was not possible to the best of our knowledge, due to the specific nature of this work.

1.4 Objective and Expected Contributions

The main objective of this thesis was to develop a semantic knowledge graph built on top of a subset of ArquivoPT's news. This extra layer's purpose is to allow for more complex queries based on entities, their relations, and the associated temporal information. It also integrates with machine learning components to attempt a more accurate representation of the world. Such a layer has the goal of allowing for a more exploratory searching mechanism than usual, while also easily enabling the analysis of data. It should be able to respond to queries from users and machines alike using a graph querying language.

By achieving this objective, this thesis is expected to provide the following contributions:

1. A flexible NLP pipeline aimed at extracting information from Portuguese texts.
2. A graph model meant to structure and store data regarding entities, relations, and temporal information.
3. A searching mechanism that combines queries and algorithms to extract relevant information or news articles from the structured data.
4. A novel evaluation method meant for automatically generated entity-relation graphs.
5. A ready-to-submit research paper that explains the previous contributions and makes them available to the scientific community.

1.5 Document Organization

The remainder of this document is organized in the following chapters:

Chapter 1 - Introduction: This chapter describes the context and motivation, as well as the objective and challenges of the developed knowledge graph.

Chapter 2 - Background and Related Work: This chapter provides the theoretical background of topics related to this thesis as well as existing techniques and graphs.

Chapter 3 - Data Acquisition: This chapter describes the method by which the data to be used was acquired from its source, ArquivoPT.

Chapter 4 - Graph Models: This chapter in turn presents the different models explored, the selected one as well as different algorithms to use the graph with.

Chapter 5 - Natural Language Pipeline: This chapter provides the specification of the generic NLP pipeline that allows the flexible creation of entity-relation graphs.

Chapter 6 - Incorporating Existing Knowledge: This chapter puts forth the ideas behind combining the data we automatically generated with Wikidata information, which can be achieved in different ways.

Chapter 7 - Evaluation and Results: This chapter presents the evaluation method, the ground-truth developed, tasks and metrics used to classify the graph, along with the results and observations obtained.

Chapter 8 - ArquiWIZ: Graph Based Web Archive Exploration: This chapter shows the developed prototype, which enables non-technical users to take advantage of some of the graph's capabilities visually.

Chapter 9 - Conclusions and Future Work: In this chapter the developed work is summed up, and a retrospective analysis is performed, introducing some avenues by which this work could be continued.

2

Background and Related Work

This chapter describes concepts, tools, and other research works in order to gain a better understanding of the theories and processes involved in this type of development. First, a description of different natural language processes is presented, along with their use for graphs and existing tools for the Portuguese language. Next, knowledge graphs are introduced, with a specification of the available data models, the different kinds of possible graphs as well as examples of existing works. Finally, web archives are described with a focus on works that develop knowledge graphs layered over them.

2.1 Natural Language Processing

Extracting information from unstructured form, natural language, to a structured format, the graph, is not a trivial matter. To this end, some form of Natural Language Processing is required, so that specific, structured information can be obtained and the graph populated with it.

Natural Language Processing, henceforth referred to as NLP, is a branch of Artificial Intelligence and Linguists, focused on allowing computers to understand the statements or words written in human language [31].

NLP allows for the birth of all kinds of systems and tasks which work with natural language, such as Machine Translation, Question-Answering, Automatic Text Summarization, Natural Language Generation, Information Retrieval, Information Extraction, and more [31].

In this context, tools of this kind will be used in a *off-the-shelf* manner. While the goal is not to improve them, it is important to be aware of the information they provide for a given text as well as their accuracy.

NLP stands at the root of the work to be developed. All information inserted into the graph will come from the outputs of these tools. Because this extraction is automatic, a certain degree of error is expected, which we will seek to minimize as much as possible through the choice of the best available combination of tools for each process.

2.1.1 Information Extraction (IE)

Information Extraction (IE for short) is the NLP high-level task that focuses on finding occurrences, in free-form text, of particular classes of objects and relationships between them [61]. For example, extract instances of storms from weather reports, with fields for temperature, wind speed, and precipitation. Because this information is extracted into a structured form, it is ideal to populate databases much like the one this thesis aims to build.

IE systems are often built with pipelines in which each step represents an NLP task, from lower to higher level tasks. The accuracy of each step is important, as errors gradually accumulate. Some works have attempted compressing multiple, related tasks (such as Named Entity Recognition and Named Entity Disambiguation) into a single one to avoid this error accumulation [45].

Parts-of-Speech (POS) tagging. Is the task of assigning words in a sentence a *tag* to a particular part of speech according to its definition and context. This means for example tagging a verb in a sentence with `VERB` and a noun with `NOUN`. POS taggers often serve as pre-processors for more abstract NLP tasks [44].

Named Entity Recognition (NER). This task's goal is to identify in natural language text **names** of entities and classify them according to their type, such as person, organization, or location [23, 80, 45]. For example, given the sentence "*Marcelo Rebelo de Sousa falou hoje em Lisboa*" ("Marcelo Rebelo de Sousa spoke in Lisbon today"), a Named Entity Extractor should extract the entities and respective types displayed in Table 2.1.

Table 2.1: Entities extracted from the sentence "Marcelo Rebelo de Sousa falou hoje em Lisboa".

Name	Type
Marcelo Rebelo de Sousa	PER
Lisboa	LOC

This task is often accompanied by Named Entity Disambiguation and Linking - which is the task of comparing the recognized entity to an existing knowledge base and from all the existing possibilities with the same name, selecting the correct one (given the context) and establishing a link between that knowledge base and the extracted entity [63]. The process of performing all three of these tasks together is often called Named Entity Extraction (NEE).

Relation Extraction (RE). Also known as Relation Detection and Characterization, this task focuses on the extraction and classification of implicit or explicit semantic relationships between entities (found previously via NER) in the text [83]. For example, a possible relation to be found is $\langle$*Marcelo Rebelo de Sousa*, *PresidentOf*, *Portugal*$\rangle$. The simplest case

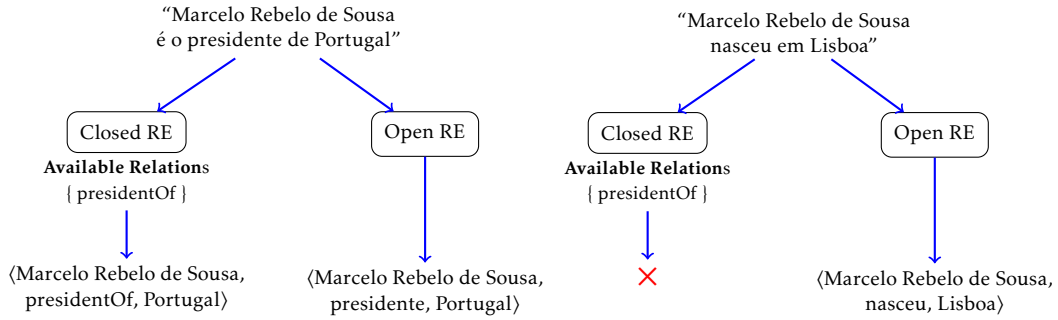


Figure 2.1: Visual example of the difference between closed and open RE.

of RE is that of binary relations, mapped to a fixed set of possibilities - this is called *closed* extraction. There is also the case where relations are not previously known or limited (particularly in situations with a lot of heterogeneous data) and are elected from the text as it is analyzed, which is called *open* extraction [17]. An example of the difference between these two kinds of REs can be found in Figure 2.1.

Semantic Role Labelling (SRL). The main goal of this task is to identify exactly what semantic relations hold between a predicate and its associated participants and properties. These relations are drawn from a previously specified list of semantic roles associated with that predicate (or class of). For example, the phrase below would be annotated as shown.

"A menina disse olá"

"$[\text{A menina}]_{\text{Agent}}$ $[\text{disse}]_{\text{Pred}}$ $[\text{olá}]_{\text{Subject}}$"

SRL also annotates the time and location relations found in the text, for each predicate [39].

Event Extraction (EE). The detection of events is considered a complex form of IE, as it involves knowledge from several different domains as well as information extracted from other NLP tasks [27]. The general view of events in this context is that given a sentence, in natural language, events denote an activity or state of action. In addition, this task finds the entities related to the event at hand: actors, places, objects, and time [54]. In practice, the extraction process aims to fill a template defining the typical information associated with that kind of event, which will often be named entities [30]. For example, given the sentence *"O president anunciou a nova lei"* (*"The president announced the new law"*) and a template `Statement` of which the fields (or slots) are `Speaker` and `Message`, an event extractor would be able to fill out this template's fields with *"presidente"* and *"a nova lei"*.

Sentiment Analysis (SA). The task of classifying the sentiment polarity from a given text is called Sentiment Analysis. It classifies a document or sentence as "positive", "negative" or "neutral" according to the tone of the text. It can also be used to infer the sentiment towards particular entities (referred to aspect-level SA) [41].

2.1.2 Semantic Similarity and Embeddings

A portion of the representation learning field (the study of different ways of representing knowledge) focuses on the transformation of text and objects into semantic meaning. An effective form of doing this transition is by converting text into embeddings. Embeddings in NLP are dense vectors, which map the meaning of more complex objects - such as words, documents, concepts, etc. - into numerical vectors. These vectors can easily be compared to each other - the closer two vectors are, the more similar the meaning of those two objects is.

Embeddings have shown to be useful in NLP and in recent years have served as the backbone for many of the previously described tasks. In addition, they have been used for semantic search, sentence similarity, document similarity, among others.

The development of these dense vectors has evolved over the years, and considering the closeness to natural language this thesis has, we present in this section different kinds of embeddings, which may be useful in later tasks.

Word Embeddings Word embeddings are low-dimensional vectors learned from text-corpora, which represent individual words [5]. Word2vec [42] and GloVe [52] are the two most prominent architectures that work over word embeddings, which calculate embeddings during training. Embeddings for these words are thus static.

Word embeddings were a first step in mapping out the semantics of words, which allowed for comparison between documents that went beyond syntactical.

Despite this, they presented a major issue, in the form of meaning conflation deficiency. Because each word is assigned a single point in space, words with multiple meanings, such as *nail*, are condensed into a single representation. This is not ideal, as the context of each word is lost, and leads to strange, incorrect proximities between words [5].

Another limitation of word embeddings is their aggregate size. Word embeddings map each individual word, which means a large document will be assigned several dozens of embeddings. While this is not a problem for smaller or more fine-grained cases, detecting the similarity between two documents often involves the calculation of its word embeddings as a cloud [32]. This technique incurs a loss of meaning in its comparison and requires a large amount of space to store all the embeddings of every document.

Deep Contextualized Word Embeddings To solve the previous issue of meaning conflation deficiency in static word embeddings, deep contextualized word embeddings were

developed. These embeddings are not static like the ones mentioned before and are calculated in real-time. The architecture takes as input a sentence and calculates for each word a representation of its meaning *within* that sentence [53].

The original architecture for calculating these dynamic embeddings was ELMo, which was based on a neural network architecture [53]. BERT presented itself as an arguably simpler way of calculating deep contextualized word embeddings while also increasing their quality by using self-attention mechanisms [72].

BERT was trained to predict following sentences, but it also allowed for the comparison of two documents between each other, by having both documents be fed into the transformer which produced a similarity score - this is what is called *cross encoding*. While the results are accurate, this does not scale well for situations with a lot of document comparisons, as every pair of documents needs to be fed into the system, which has to then re-compute the embeddings for that specific pair [58].

These attention-based models are often referred to as transformers. They generate embeddings not for words, but for tokens. These tokens are sequences of characters that are learned during training. Word in text are broken down into their composing tokens - for example, the word "Presidente" might split into *_Pres*, *i*, *dente* (the _ represents a token that cannot suffix another token). Each token is assigned an embedding, and getting the embedding of a word is often accomplished by averaging the embeddings of its composing tokens.

2.1.3 Uses of NLP for Graphs

As previously mentioned, NLP allows the transformation of unstructured information in the form of text to specific structured information. The exact type of information extracted depends on the task being used.

Similarly, the information to be extracted from texts depends on the goal of the graph to be built. As such, it is necessary to first understand how NLP can be used to construct graphs. Table 2.2 offers a summary of the previous section, along with some possible applied uses for the tasks at hand in the context of graphs.

2.2 Knowledge Graphs

Colloquially, knowledge graphs are a subset of graphs that allow for searching mechanisms other than keyword-based. Specific concepts such as the aforementioned entities and events can be structured and connected, giving the user the possibility of moving through these connections to find the right information.

The term knowledge graph has existed for years, with mentions of the topic dating as far back as 1973 [62]. A more recent and notable example is Google's Knowledge Graph [65], which stores information regarding different entities and facts to quickly describe search items. Despite this, no universal definition for what *exactly* is a knowledge

Table 2.2: Possible uses for previously described NLP tasks in the construction of knowledge graphs.

NLP Task	Information Obtained	Use for Graphs
Part-of-Speech Tagging	Assigns tags to each word in a sentence according to their part of speech (Noun, Verb, etc.)	Used mostly for pre-processing
Named Entity Extraction	Finds names of entities (people, locations, etc.)	Names can be converted into nodes of the graph
(Binary) Relation Extraction	Given two entities, finds in the text their relation	Relations can create edges between nodes of those entities
Semantic Role Labelling	Tag relations between predicates and participants. This allows the extraction of more fine-grained, sentence-level events	These events can be added to the graph through new nodes that connect to all participants
Event Extraction	Fill event templates with information about the event (participants, time, location, etc.)	These events can be added to the graph through new nodes which connect to all participants
Sentiment Analysis	Infer polarity of sentiment from a document (positive, negative or neutral)	May be added to the node of a given text as a property.
Sentence Embeddings	Similarity score between documents	Can be used to connect seemingly unrelated nodes referring to articles

graph has been defined [4, 16] (Hogan et al. [26] provide a discussion of the evolution of definitions over the years). In this thesis, the adopted definition is the one provided by [4], which defines knowledge graphs as:

Definition 2.2.1 (Knowledge Graph)**.**
A graph of data with the intent to compose knowledge

While seemingly simple, this definition contrasts with others that force knowledge graphs into a particular type of data structure [50], explicitly state that it must have reasoning capabilities [16] or that restrict the type of its nodes [26].

Brattka et al. [4] define "composing knowledge" as the "continual process of extracting and representing knowledge in a manner that enhances the interpretability of the resulting Knowledge Graph" but allow alternative definitions to avoid too strong of a restriction on what a knowledge graph is. In the case of this thesis, one could argue the incremental addition of more news articles to the graph satisfies this definition.

2.2.1 Graph Data Models

A **data model** is a set of conceptual tools which describe the data, its relations and semantics as well as its constraints (or lack thereof) [64]. Essentially, a data model is how the user *views* the data. This section presents the different kinds of data models commonly available to describe graphs, as well as the choice of the most appropriate one for the proposed research hypothesis.

Data models make no reference to how the data is stored, as that is a lower level of abstraction (called the physical level) [64]. The choice of a good physical level is important, as the way information is stored on disk speeds up or slows down certain queries. It usually depends on the database management system (DBMS) used. With this in mind, the selected DBMS should be one that efficiently responds to the expected type of queries while following the selected model.

The definitions that follow, are agnostic to *types* or implementation details purposefully. They are meant to serve as general theoretical definitions to help decide the best model choice in regard to what they can accomplish and how the stored data is represented.

Directed Edge-labelled Graphs Directed edge-labeled graphs (edge-labeled graphs or del graphs for short) are simple and widely adopted [1]. This model is composed of nodes and edges, which are labeled. Edges and their labels thus indicate the type of relationship between two nodes. Directed edge-labeled graphs are formally defined in Definition 2.2.2.

Definition 2.2.2 (Directed edge-labeled graph)**.** A del graph G is a pair (V, E) where

1. V is a finite set of vertices (or nodes)
2. E is a finite set of edges, where $E \subseteq V \times Lab \times V$ and Lab is a set of labels.

Following this definition, an example instance of an edge-labeled graph is formally presented. Its visual representation can be found in Figure 2.2.

$$V = \{\text{Marcelo Rebelo de Sousa, Person, Country, Portugal, City, Lisbon, 12 December 1948}\}$$
$$Lab = \{\text{type, president\_of, capital\_of, born}\}$$
$$E = \{(\text{Marcelo Rebelo de Sousa, } \textit{type}\text{, Person}), (\text{Lisbon, } \textit{type}\text{, City}),$$
$$(\text{Marcelo Rebelo de Sousa, } \textit{born}\text{, 12 December 1948}), (\text{Portugal, } \textit{type}\text{, Country}),$$
$$(\text{Marcelo Rebelo de Sousa, } \textit{president\_of}\text{, Portugal}), (\text{Lisbon, } \textit{capital\_of}\text{, Portugal})\}$$

A real-world use case of directed edge-labeled graphs can be found in the Resource Description Framework (RDF), the W3C standard recommendation for representing and exchanging knowledge in a machine-readable way through the Web [57]. RDF graphs are defined as a set of triples analogous to the set E and its nodes, set V, which can be split

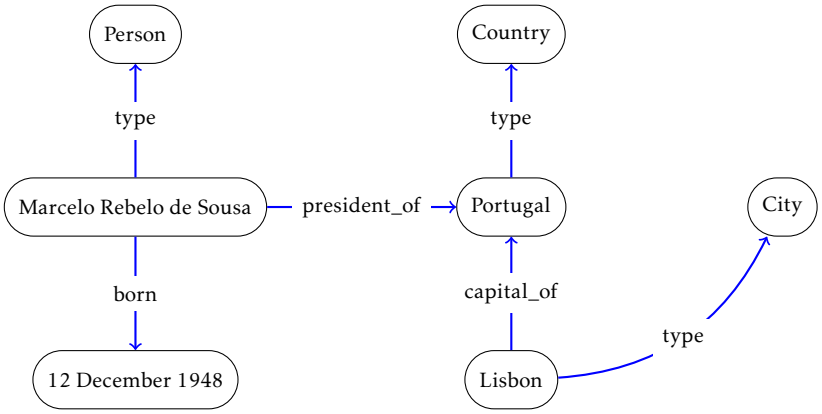


Figure 2.2: Example of a directed edge-labeled graph.

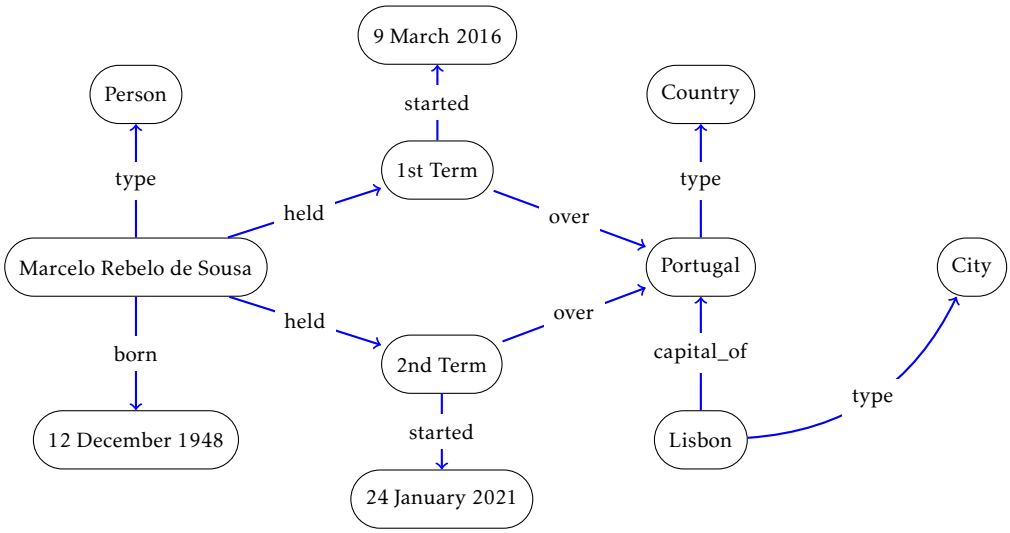


Figure 2.3: Example of a directed edge-labeled graph with n-ary relationships.

into IRIs (which point to *resources*), literals (such as strings, integers, etc.) or blank nodes. In RDF, *Lab* is restricted to IRIs which can also be referenced in nodes.

There are two important aspects to take note of in del graphs: edges do not have attributes and E is a set. So if one wanted to have a *president_of* edge for each term of office, it would be impossible to have two *president_of* edges between Marcelo Rebelo de Sousa and Portugal, nor could the date of the term be assigned to the edge. Workarounds for the addition of n-ary (where $n > 2$) relationships are possible, with the addition of "extra" nodes, as exemplified in Figure 2.3. This generates a certain friction in the use of the graph, as its structure is no longer as intuitive.

Heterogeneous Graphs Del graphs are flexible, but by default, their nodes have no semantic meaning. In the previous examples (Figures 2.2 and 2.3) this was explicitly added through the use of *type* relations, despite not being mandatory. Their addition provides a better understanding and subsequent use of the graph however, which led to the creation of the heterogeneous graph model.

Heterogeneous graphs are similar to directed edge-labeled graphs, but where nodes are explicitly labeled thus indicating their type without the need for extra nodes and relations [68, 81]. This graph model is formally defined in Definition 2.2.3.

Definition 2.2.3 (Heterogeneous graph)**.** A heterogeneous graph G is a pair (V, E) with two object mapping functions ϕ and ψ where:

1. V is a finite set of vertices (or nodes)
2. E is a finite set of edges
3. $\phi : V \rightarrow Lab_N$, which given a node, returns its label. Lab_N is the set of node labels
4. $\psi : E \rightarrow Lab_E$, which given an edge, returns its label. Lab_E is the set of edge labels
5. $|Lab_N| > 1$ and $|Lab_E| > 1$

A graph where $|Lab_N| \leq 1$ and $|Lab_E| \leq 1$ is called a **homogeneous** graph because all information pertains to a single domain/label [68]. Heterogeneous graphs are thus limited to a single label per node, which was not the case with del-graphs where there could be multiple outgoing *type* edges. Adapting Figure 2.2's example into a heterogeneous graph is simple, and is shown in Figure 2.4.

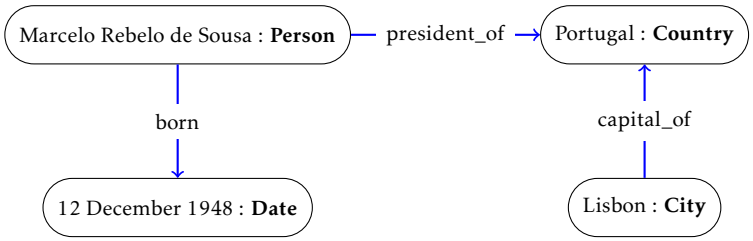


Figure 2.4: Example of a heterogeneous graph.

Labelled Property Graphs As seen before, adding information about edges to a del-graph (or a heterogeneous graph) is cumbersome, as the previous edge needs to be replaced by a node and several more edges added for each attribute. If this is done late in the graph construction, it might force a massive reorganization of many edges - which means it is necessary to decide whether a certain edge will have attributes beforehand. It also makes the graph harder to analyze, as relations that a user might consider to be edges are actually presented as nodes.

For situations where it is necessary to add information to edges or regularly update data in the graph's edges and nodes, the use of property graphs is recommended [1].

Labeled property graphs are composed of nodes, relationships, properties, and labels. Nodes contain properties, their attributes essentially, and can be tagged with labels, which group together nodes according to their semantic meaning - similarly to the heterogeneous graph labels. Relationships connect nodes and have a direction, thus providing

the graph with structure. Similarly to nodes, relationships can be assigned properties and have a single *type* (or label) [76, 1].

An important distinction between property graphs and the other models is that in the former, nodes and edges have unique identifiers outside of their values. This means that repeated nodes (nodes with the same values/attributes) and edges (edges with the same origin and destination) are possible.

A formal definition for labeled property graphs can be found in Definition 2.2.4. The labeled property graph is the model adopted by graph databases such as Neo4J [76].

Definition 2.2.4 (Labeled property graph)**.** A labeled property graph G is a tuple $(V, E, \rho, \lambda, \theta, \sigma)$ where:

1. V is a finite set of vertex (or node) ids
2. E is a finite set of edge ids such that V and E are disjoint
3. $\rho : E \to (V \times V)$, is a total function (every edge in E must have a return value). This function defines the origin and destination of an edge: $\rho(e) = (v_1, v_2)$ indicates that e is a directed edge from v_1 to v_2.
4. $\lambda : V \to \wp(Lab)$ is a total function with $\wp$ representing the power set (power set of a set is the set of all its subsets, meaning any combination of elements of Lab can be returned). Lab is a finite set of node labels. This function defines the set of labels (including the empty set) of a node: $\lambda(v) = \{Lab_1, Lab_2\}$ indicates that node v has labels Lab_1 and Lab_2.
5. $\theta : E \to Type$ is a total function. $Type$ is a finite set of relationship types. This function defines the type of an edge: $\theta(e) = t_1$ indicates that edge e has type t_1.
6. $\sigma : (E \cup V) \times Prop \to Val$ is a partial function. $Prop$ is a finite set of properties and Val a set of values. It defines the value of a property on a given node or edge: $\sigma(e, \text{date}) = 5/2/2000$ indicates that edge e has a property *date* with the value *5/2/2000*.

An example equivalent to the one in Figure 2.3 can be found in Figure 2.5.

Of all three presented models, the one chosen to organize the developed graph was the **directed edge-labeled graph**. This choice comes from its adhering to the RDF standard, which in turn also allows linking to other knowledge bases and easy incorporation of existing facts. In addition, changes in the graph will not be common, ruling out one of the main advantages of the labeled property graph.

2.2.2 Knowledge Graph Types

The previous discussion about the ways data can be modeled into a graph was agnostic to its semantics. By taking a higher-level look at the *meaning* behind the data stored in a

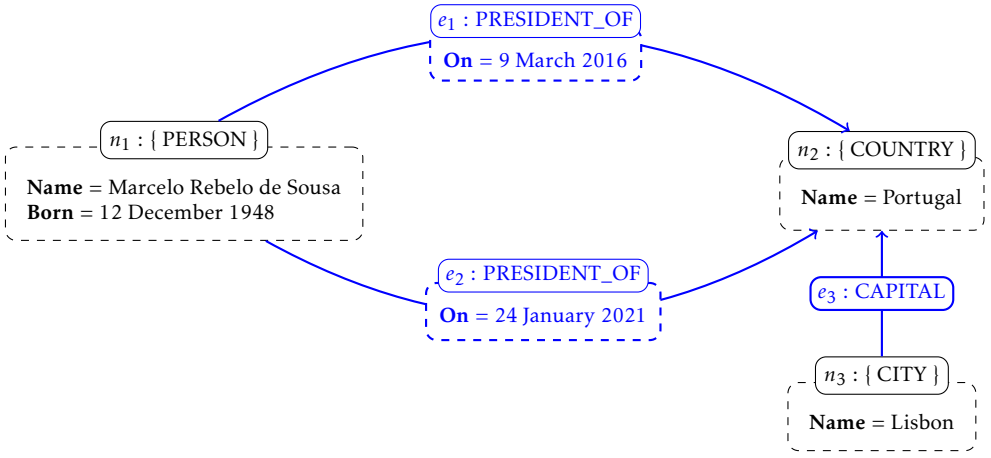


Figure 2.5: Example of a labeled property graph.

graph, it is possible to distinguish knowledge graphs based on their focus - the semantic significance behind their nodes and connections. Considering the scope and context of this thesis - to model a semantic graph over a web archive - three possible high-level types of graphs are defined here, which will later be used to classify existing works.

Entity-Centric. Knowledge graphs that focus on describing entities of interest (people, locations, organizations, etc.) and the relations between them are defined here as Entity-Centric KGs, however in most literature are simply defined as regular Knowledge Graphs [26]. These graphs often store facts as static, dismissing temporal validity [20, 79].

Temporal Entity-Centric. Information about entities is rarely static - facts and relations change over time. To accommodate this, some knowledge graphs include time intervals and temporal information into their edges to define the validity of that relation. These knowledge graphs are usually called Temporal Knowledge Graphs [20, 79], and are here referred to as Temporal Entity-Centric because their main goal is to define *temporal* facts over entities.

Event-Centric. Events can be defined as anything that happens at a certain point in time. Knowledge graphs in which all stored information is about events are called Event-Centric Knowledge Graphs [59]. Because events naturally involve time (when it started and when it ended for example), these graphs have a strong temporal aspect to them. They also include entities and some relations between them - these entities are usually participants of the described events (location where it happened, people and organizations involved, etc.).

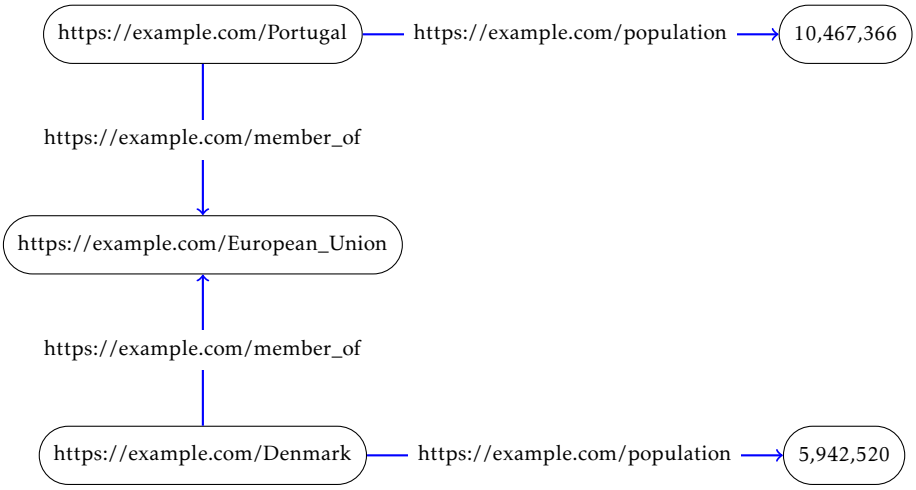


Figure 2.6: Example graph to demonstrate SPARQL query. As an RDF graph, its edges must be URIs as well as its nodes, which can also be literals or blank.

2.2.3 Graph Querying

Knowledge graphs provide data with semantic structure, which allows the user to search through the nodes and edges according to their meaning. The best way to allow this traversal of a graph is through a graph query language, which enables querying the graph for information. To fully understand what kind of questions can be answered and what information is possible to extract, it is necessary to know the capabilities of the language.

The official language for querying RDF graphs is SPARQL, which supports several different features for manipulating or inspecting graphs. SPARQL is based on pattern matching to respond to queries. A *graph pattern* is a graph-structured query that will be matched against the database. They are essentially graphs, following the same model being queried, with variables. Matching this pattern means finding all sub-graphs of the database where these variables are mapped to actual values [1].

Since RDF graphs are represented as triples - subject, predicate, object - SPARQL takes a similar approach, where the triples added to the query must be matched. Any part of the triple may instead be replaced by a variable, which the DBMS will fit to any matching value. For example, the query shown in Listing 1 querying the graph in Figure 2.6 is requesting all the possible values of `?country` which are members of the European Union and have at least 10 million inhabitants. The answer in this case should be `<https://example.com/Portugal>`.

Similarly to SQL, SPARQL supports filtering, set logic, and aggregation among other complex query operations. Of particular importance are *property paths* which allow SPARQL to perform complicated pattern operations, similarly to regular expressions. For example, if the predicate of the triple in the query was `ex:member_of*`, the DBMS would replace `?country` with any node that has a path composed only of `ex:member_of` relations to the European Union, instead of only those directly connected to it.

Property paths are one of the most powerful features in SPARQL, but by default, they

```
PREFIX ex: <https://example.com/>

SELECT ?country
WHERE {
    ?country ex:member_of ex:European_Union .
    ?country ex:population ?population .
    FILTER ( ?population >= 10000000 }
}
```

Listing 1: SPARQL query example, requesting all the countries that are members of the European Union and have a population of at least 10 million.

are binary operations. A property path is checked to be true or false and does not allow for access to its intermediate nodes. While for most use cases this is likely fine, it makes it impossible to apply constraints to the path chosen. Some DMBSs actually implement pathfinding as its own service within the system, such as GraphDB[1], which allows for more control over the path itself.

2.2.4 Graph Knowledge Structuring

When referring to knowledge graphs (and knowledge in general) it is common for the term *ontology* to appear. Ontology (with an uppercase O) is a philosophical discipline which deals with the nature and the organization of reality [9]. However, *an* ontology (lower capital o) in engineering is used in different contexts and often ambiguously. Chandrasekaran et al. [9] define an ontology as a representation vocabulary that provides terms with which to describe facts of a certain domain. A more concrete definition, which complements the previous one, is that "an ontology is a formal, explicit specification of a shared conceptualization" [66] which adds the idea that ontologies are meant to be *shared*.

W3C adopt a more pragmatic definition[2], specifying that ontologies define the concepts and relationships (“terms”) being modeled as well as possible constraints between them. Ontologies also represent the building blocks for inference techniques. Well-defined ontologies allow for the sharing of knowledge of a certain domain since it can be used in any instance inside it. This means other people wanting to model information from the same domain can adopt the ontology, making the data representation uniform across systems as well as allowing them to skip the domain-analysis step [9].

Similarly, database schemas represent the logical design of a specific database [64]. In essence, schemas and ontologies are the same thing: ways to model the data (the knowledge) of a certain domain, whereas ontologies are a broader term, used across disciplines.

There exist many examples of knowledge graph ontologies, both general purpose

[1] https://graphdb.ontotext.com/
[2] https://www.w3.org/standards/semanticweb/ontology

(RDFS[3]/OWL[4]) and for specific situations (for example, FOAF[5] is used to describe people and social relationships on the Web). These ontologies are most often defined and used in RDF, but their adaptation into other models is possible (such as by Carnaz et al. [8]). In general, due to the shared nature of ontologies and their domain accuracy, there exist works whose sole focus is on developing standard-adhering ontologies.

2.2.5 Knowledge Graph Examples

Knowledge graphs have increased in popularity over the years, and there is no shortage of them in the wild. The way a knowledge graph is constructed depends on a variety of factors such as the conceived applications, the available sources, and the domain itself.

In general, knowledge graphs are constructed by extracting information from various data sources and providing it with structure. Variations in this method occur in regard to the type of data source (natural language, relational databases, human knowledge, etc.), the extracted information (entities, relations, domain-specific, etc.), and how the extraction happens (NLP processes, by hand, specific parsers, etc.).

Over the course of this section, several existing works are presented, along with their developed knowledge graphs and construction methodology. A summary of all examined works and more can be found in Table 2.3.

DBPedia [33] is a large, open enterprise knowledge graph. It describes information automatically extracted from Wikipedia pages in over 100 languages. Its largest language section, English, consists of over 400 million facts that describe 3.7 million things. DBPedia is defined in RDF and its contents are parsed from info-tables in Wikipedia pages. Wikipedia pages have recommended (pre-defined) and custom tables in which writers can insert information about the topic at hand. DBPedia has hand-written mappings which convert recommended tables into facts and a generic parser for user-defined tables. DBPedia publishes its information in well-known RDF schemas and links its facts to other data sets, heavily contributing to the Linked Open Data cloud.

The SEMCrime Framework [8] enables the lexical, syntactic, and semantic analysis of criminal-related documents, which populates a Neo4J database. Its input varies from online crime newspapers to police reports. These documents are pre-processed before being fed into an NLP pipeline, which uses an extended NER to extract entities as well as crime-related terms such as narcotics and crime types. Events are in turn extracted through SRL to identify the 5Ws and 1H[6] described in documents. Data is exported into an XML before being loaded into the database, the schema of which is based on the simple event model [24]. The framework also connects to an external geographic database for extra enrichment.

[3] https://www.w3.org/2001/sw/wiki/RDFS

[4] https://www.w3.org/TR/owl2-rdf-based-semantics/

[5] http://xmlns.com/foaf/0.1/

[6] *Where*, *When*, *Who*, *What*, *Why* and *How*. Simple strategy used by investigative journalists, detectives, and researchers to collect a complete and accurate story.

SM Entity Tracker [19] is a knowledge graph built between entities with the goal of allowing the analysis of public opinions towards them as well as how strongly these entities were connected to each other. In this work, a large set of Twitter posts is used as the data source. Entities are extracted from posts, as well as the sentiment behind the posts - which becomes associated with the extracted entities. Entities are assigned scores, such as popularity, according to the sentiment values returned and the time periods of posts in general. Entities are connected to each other through coreference in posts. Although theoretically, it is a knowledge graph, the data is structured as `CSV` files, and node/edge calculations are done dynamically.

NewsReader [59] is a system that "reads" news and describes what happened, who is involved, where, and when as a graph. From a dataset of news, an RDF graph is constructed which describes the events mentioned in the documents, in four different languages, with a focus on English. The graph is fine-grained: each entity and event point toward the section of the text in which they were referenced. A complex NLP pipeline was built in modules and includes entity extraction, entity disambiguation, opinion mining, semantic role labeling, among others. The extracted events are passed through a co-reference module, to identify which news articles reference the same events.

Although only a handful of existing graphs are presented here, these highlight a set of available strategies for constructing knowledge graphs, some of which can be adjusted for the purpose of this thesis.

2.2.6 Knowledge Graph Embeddings

The most common use for knowledge graphs is as a semantic database with a powerful querying mechanism and inference system. However, the heterogeneous nature of triples and the naturally large size of knowledge graphs make them hard to use at times, which led to the research of other ways to take advantage of their semantic structure. One such way that has been gaining traction in recent years is the concept of Knowledge Graph Embeddings.

Knowledge Graph Embeddings are continuous space vectors, which represent the entities and relations of the graph - similarly to word embeddings, but for entities and relations. This simplifies their overall format while maintaining the structure of the graph [79]. These embeddings are usually generated through learned models and have been used in a multitude of downstream tasks such as (temporal) knowledge graph completion [79], question answering [29], and many others.

Knowledge graph embeddings are capable of providing semantic embeddings for entities and relations, which allows them to be useful in any context that focuses on entities and their relations. Another possible use of knowledge graph embeddings is when they are combined with sentence embeddings to combine existing knowledge and textual semantics [48].

Due to the evolution of facts over time in knowledge bases, some works have developed embeddings models which take time into account. This means that the temporal aspect directly affects the semantics of entities.

2.2.7 Knowledge Graph Evaluation

Independently from its sources of data, a knowledge graph will likely be incomplete, and possibly contain contradicting information. Evaluating the quality of a knowledge graph after its completion is thus essential. When referring to the quality of a knowledge system it is often defined as *fitness for purpose*, as in, the quality of that system is measured by how well it performs its proposed tasks.

Completeness and Semantic accuracy In knowledge graphs, there exist many different metrics by which one can evaluate the database [26]. Of particular note are completeness and semantic accuracy. Completeness refers to how much of the world (or score that is being modeled) is actually represented in the data while semantic accuracy refers to how correct that knowledge is.

In any knowledge system not focused on a particular domain, completeness is nearly impossible to obtain - the world is too vast to possibly contain in a single database. Semantic accuracy, however, is more realistic, as the existing data need only be correct not necessarily complete.

Fitness for purpose can often be reduced to semantic accuracy. Knowledge systems are focused on allowing the search, discovery, and analysis of knowledge. For any of these to be correctly performed, that knowledge needs to be correct to an extent.

Methodologies While semantic accuracy may be easier to attain than completeness, it still needs to be verified, especially in the context of automatically generated graphs. Ideally, a knowledge graph's semantic accuracy would be evaluated by comparing the system to a *gold standard*, a dataset in which we know the data to be correct. However, such datasets usually do not exist and are hard to create, which leads to the usage of different methodologies in different contexts.

Some works opt for simply verifying a few use cases of their graph and checking whether they make sense [19]. A more rigorous version of this type of evaluation is by taking a large portion of the generated triples and manually checking whether they are correct according to some metric [59].

Because verifying the accuracy of a graph is a complicated task, it is also possible to instead measure how well that created system compares to existing ones through a user study [18]. Such a methodology is valid in the context of Fafalios et al. [18] because no new knowledge is produced (simply retrieved from Wikidata), however in general it does not equate to semantic accuracy, even if the study demonstrates good results.

Other works skip evaluating the resulting graph and focus on verifying how correct the NLP pipeline is. This is done by evaluating each of its modules independently and taking their overall accuracy as a suitable metric for the graph [8]. While easier, this approach disregards the complexity associated with the structuring of a graph as well as the error accumulation in the sequence of modules - two modules working with 50% accuracy does not mean the resulting graph will have 50% accuracy, but instead much lower.

Knowledge graph completion [79] is an area of research that focuses on completing graphs with possibly missing triples, often using knowledge graph embeddings, also often referred to as the task of *link prediction*. Works in this area usually follow a task of evaluation which takes in a dataset of triples (such as ⟨*Marcelo Rebelo de Sousa*, *PresidentOf*, *Portugal*⟩), and masks either its subject or object. The system is then asked to predict the masked part of the triple and the score is calculated based on how correct its predictions are. A similar approach can also be adopted for evaluating automatically generated graphs.

2.3 Web Archives

Information spread across the web is ephemeral by nature: in 2004, it was estimated that after a year, 80% of links on the web were replaced by new ones [47]. In order to fight this loss of information, web archives were created. Web archives are a special kind of digital libraries that preserve all sorts of digital information, including videos, images, and even digitized documents [11].

Nowadays there exist dozens of web archive initiatives, with one of the oldest ones being the Internet Archive[7], which started in 1996. ArquivoPT[8] is, in turn, the largest Portuguese web archive, which contains data from as far back as 1991.

2.3.1 Searching in Web Archives

While the goal is to preserve, the data stored in web archives must also be accessible and searchable. Web archives support multiple types of searching mechanisms such as URL, domain, media-type, and keyword-based as well as filtering the results by time ranges [46, 13, 11].

Full-text searching was found to be the most desirable search mechanism in web archives, and the most used when present [12]. However, this type of search often leads to poor results [77, 18]. Keywords are particularly ineffective in the context of *exploratory searching* (when the user needs to iterate repeatedly over the data and learn about it before effectively being able to search) [37] or when the user is unsure of the contents of the collections [78].

[7] https://archive.org/
[8] https://arquivo.pt/

Table 2.3: Summary of (some) existing knowledge graphs. References marked with † are those that directly aid the searching of web archives.

Reference	Type	Source	Process
Lehmann et al. [33]	Entity	Wikipedia tables	Rule-based, specific parsers
Vrandečić et al. [73]	Entity	Wikipedia	Crowdsourced
Suchanek et al. [67]	Entity	Wikipedia and WordNet	Rule-based, specific parsers
Fafalios et al. [19]	Temporal	Twitter posts	NER, NEL, Sentiment Analysis
Carnaz et al. [8]	Event	Crime related documents	(Extended) NER, SRL
Rospocher et al. [59]	Event	News	NER, RE, SRL, and more
Cheatham et al. [10]	Temporal	Geoscience databases	Data dumps already structured
Gottschalk et al. [22]	Event	Other LOD KGs	RE, Temporal extraction
Lieber et al. [34]†	Entity	Social media posts	NER
Fafalios et al. [18]†	Entity	New York Times Corpus	NER

Full-text mechanisms do not look at the semantic relations between the documents they index, focusing on co-reference, and thus the goal of this thesis is to enhance searching by introducing *semantic relations* into the way search is performed, through the use of a graph. In addition, this graph allows the user to navigate across documents without having to rely on hyperlinks between them, but instead according to the meaning of the information they hold, that can be naturally encoded in a graph-like structure.

2.3.2 Graph-based Information Extraction from Archival Data

There has been a considerable amount of work in creating better search methods for web archives (such as Lin et al. [35]), some of which involved *navigational* graphs (the edges between documents were formed by *hyperlinks* connecting them) [28] or used event-based searching and document similarity [21]. Fewer works try to combine knowledge (semantic) graphs and web archives for searching capabilities, which is what this thesis intends to do. Some works developed in this scope are presented in this section.

The BESOCIAL workflow [34] is a methodology designed to enhance the preservation of social media posts. It functions as a web archive, which extracts social media posts from several providers and builds an RDF knowledge graph *during* the extraction process. This knowledge graph is composed mostly of metadata from the posts, but NER is performed over them as they are introduced into the archive. The entities extracted from this process are disambiguated and thus, posts referencing the same entity point to the same node. Entities are also linked to their respective DBPedia entries.

LayeredNYTimes [18] is a "semantic layer", proposed to aid in the search of web archive documents. This semantic layer is essentially an RDF knowledge graph built over the archive, without storing any of its files. This knowledge graph is composed of metadata already present in the archive files and from entities extracted from the documents

through NER, which are once again linked to DBPedia. Entities are associated to documents with confidence scores, which indicate the probability of that entity actually being mentioned; this allows the user to control their search queries' finesse. It also connects versions of the same documents, ordered according to their addition to the archive. The presented evaluation results show promise in the concept of using an added layer for search relevance.

CALMA [49] builds several layers of RDF graphs over a web archive, each with a different purpose, which can be used in conjunction or separately. However, this particular library is based on the Internet Archive's Live Music Archive and not textual documents or web pages, thus it falls out of the scope of this thesis - it serves to show that the concept of knowledge graphs layered over archives to enhance search is feasible in more than one modality.

While web archives have not been deeply explored in combination with knowledge graphs, there have been several works focused on bettering the search and accessibility of "classic" physical archives through the use of linked data technologies [25]. These endeavours are particularly focused on re-structuring bibliographic and archival data as a graph, to allow linking to other data sets (to provide extra information about its contents) and to improve the searching mechanism of the archive [14, 60]. Few of these projects (such as Rademaker et al. [55]) extract additional knowledge from the actual archived data. Thus, most of the research in these cases goes into developing flexible ontologies that correctly model information, while conforming to open standards and archival specifications.

For example, in Rademaker et al. [55] the existing metadata from the available resources was automatically extracted into an RDF graph to allow better searching as well as data sharing. Because the archive is accompanied by a comprehensive dictionary, the authors discuss the possibility of further enhancing the knowledge graph with extracted entities and a connection to WordNet [43] - a lexical database which connects words according to their semantic meaning (senses). In addition to such NLP processes, because of the multimodal nature of the archive, face recognition for photographs and audio processing for audio files is also an available route to extract more information.

Through these works, it is possible to conclude that knowledge graphs layered over web archives have a positive impact on a user's search capabilities when centered around entities, which solidifies the viability of the proposed research hypothesis.

3

Data Acquisition

News articles, archived in ArquivoPT, stand at the foundation of this thesis and its envisioned graph. The goal is to focus on a subset of reliable news, that span several years while still remaining large in size. Thus, a proper selection strategy must be defined, along with the extraction and scraping of the data itself.

3.1 ArquivoPT Collections

ArquivoPT is an extensive collection of the web, storing over 17000 million web pages from all types of sources - not all of which are reliable. A possible solution to this was to define a set of trustworthy domains to use as a filter, thus keeping only pages known to be from reliable sources. Fortunately, that work has already been partly developed by the ArquivoPT team.

The data archived in Arquivo.pt is split into collections[1]. Some of these collections were donated to the archive, while some others group pages of the same type together. Most collections however are defined by a set of URLs from which the crawler starts at certain time periods. One such type of collection are the **FAWP**, which aggregate specific news media websites, websites with frequent renewals of content and websites from government and public bodies. These crawls happen daily and are published every three months. **AWP** stands for "**A**rquivo da **W**eb **P**ortuguesa", while **F** is just an identifier.

At the time of generating this dataset, the available collections of this type spanned from **FAWP1** up to **FAWP43**. Because these collections were not restricted to news websites, specific domains (`dn.pt`, `observador.pt`, `expresso.pt` along with `publico.pt` and many of its sub-domains) would later be used to filter out the URLs of the collection, thus selecting only news pages. The full list of the domains used can be found in Annex I.

[1] `https://arquivo.pt/collections`

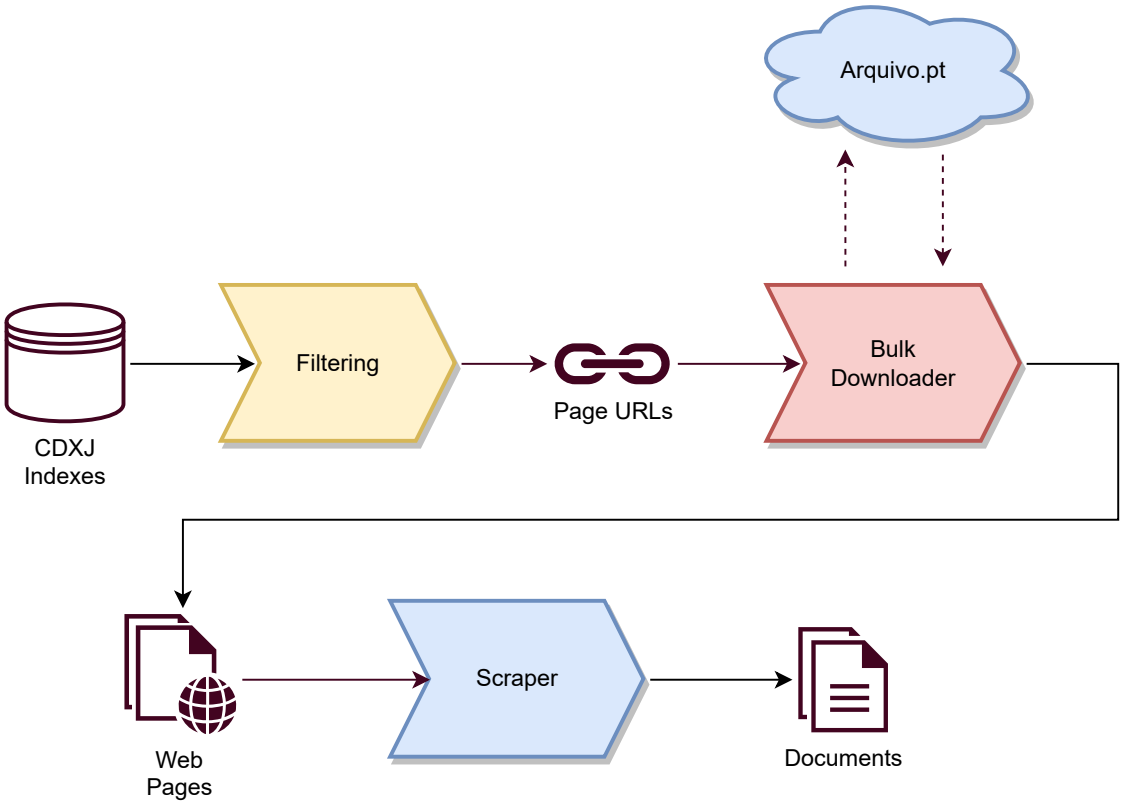


Figure 3.1: Diagram representing the article extractor part of the knowledge graph building process.

3.2 Acquisition and Scraping

Although conceptually the URLs to be used had been decided, there was still the matter of acquiring them and their respective pages. At the time, the process suggested by ArquivoPT was to use their search API[2], which allowed filtering through domain and collection. However, that search was limited to 2000 returned responses at a time, which then had to be individually retrieved anyway. This would introduce traction in an already large task.

In conjunction with the ArquivoPT team, a better technique was developed (a diagram of it can be found in Figure 3.1), which was based on the **CDXJ** indexes provided by them. These indexes contained the URL and metadata of every page, split into their respective collections. Using these indexes allowed direct retrieval of the pages themselves, while domain and collection filtering was done locally. This technique eventually led to the development of ArquivoPT's bulk API[3].

Upon downloading the websites - an example of which can be found in Figure 3.2-, the

[2] https://github.com/arquivo/pwa-technologies/wiki/Arquivo.pt-API
[3] https://github.com/arquivo/pwa-technologies/wiki/APIs#bulk

Figure 3.2: Screenshot from Arquivo.PT depicting one of the sampled news article's web page. The sections highlighted in blue dots represent those that were extracted from the site with Newspaper3k.

resulting HTML files were fed into a Newspaper3k[4] Python script. Newspaper3k is an open-source library that allows the high-quality scraping of news articles. With this library, it was possible to extract important textual data, such as the title, the date, the body, and the authors of each article. Its extraction is not perfect, however, particularly in the case of dates, as it would at times detect incorrect values - likely due to its use of regular expressions. For example, this article from January 11th of 2011 was marked as being from 23rd of September, 2099 (this seemed to happen due to the *20991123* identifier on its URL). This is an exception however, as from a sampled manual evaluation it seems to detect mostly correct dates, such as for this article where it correctly detected its publish date as 16th of November, 2019. Articles with impossible dates (such as in the future or too far in the past) had their publish dates removed and marked as if the date had not been extracted.

3.3 Final Dataset

Upon acquiring and scraping the data from all the selected domains within the **FAWP** collections, the extracted information was combined with metadata found in the **CDX** indexes (such as crawl time).

As a result, the final dataset had 2.076.829 news articles from trustworthy domains, which we named **Full Dataset**. Of those, 984.120 pages had valid published dates, spanning from the year 1991 all the way to 2020, which we named **Temporal Dataset**. The distribution of articles across domains in these datasets can be found in Figure 3.3 and it

[4] https://newspaper.readthedocs.io/en/latest/

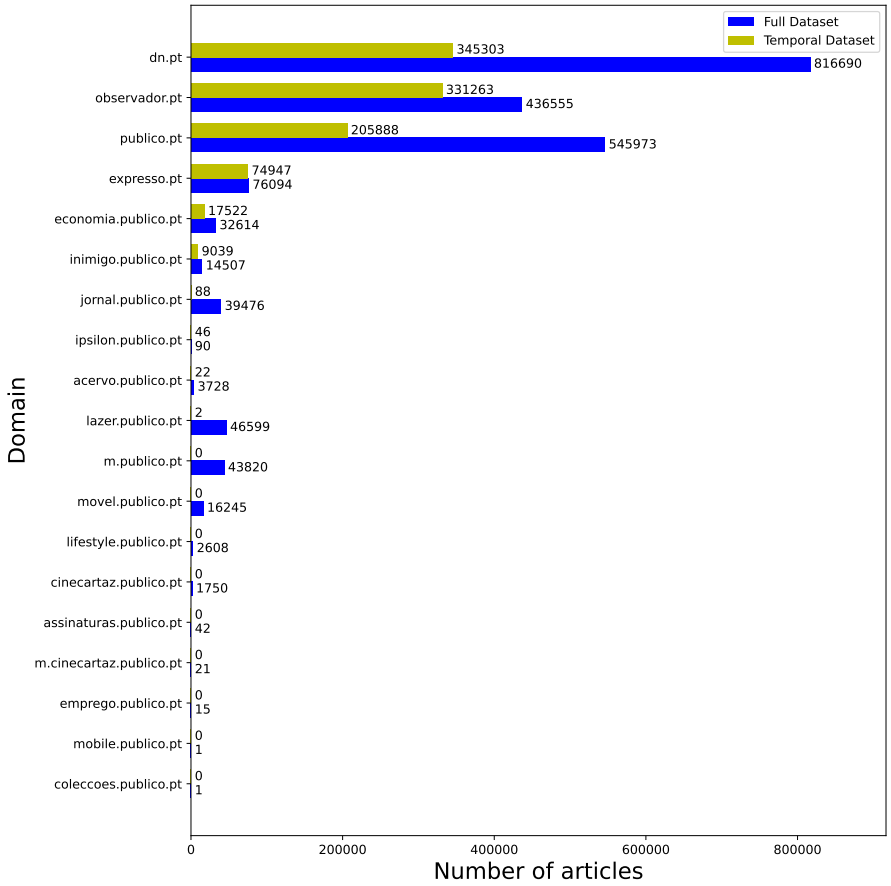


Figure 3.3: Distribution of articles across different domains in the **Full** and **Temporal** datasets.

is possible to conclude most of the articles in either one are from either *Diario de Noticias*, *Observador* or *Publico*. The distribution of articles across time (in the Temporal Dataset) grouped by year can be seen in Figure 3.4, which shows that despite a large temporal span in total, most articles were published between 2015 and 2020, which means the resulting graph will have more information regarding those years.

Due to the temporal nature of the graph, only the articles in the Temporal Dataset (meaning only articles with valid dates) were later fed into the NLP pipeline and served to create the graph.

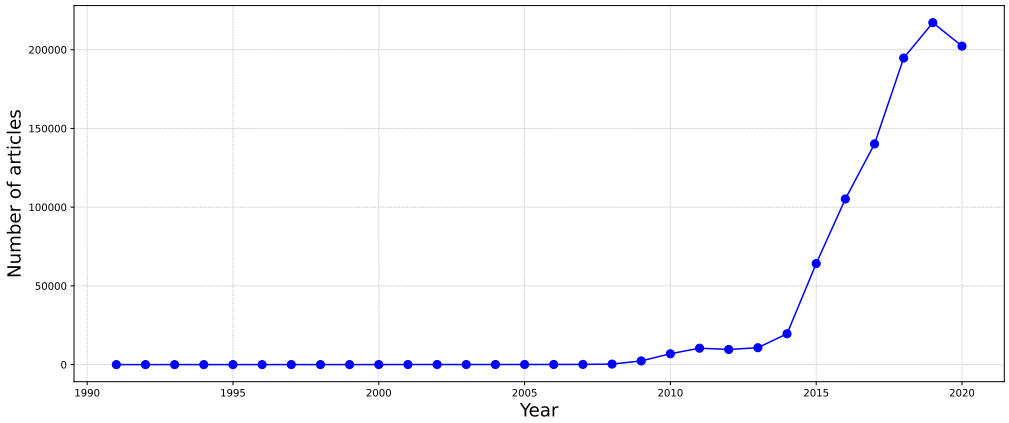


Figure 3.4: Distribution of articles across time in the **Temporal** dataset. All years with a spot have a non-zero amount of articles.

4

Graph Models

A focal point of this thesis was the development or adaptation of a graph model capable of expressing the desired semantics in a simple but versatile way. As such, several different models were developed and analyzed with the goal of understanding which one was most suited for this particular context - Portuguese texts, a vast number of articles, and a temporal component.

4.1 Preliminary Models

Before arriving at the resulting model for the graph, other conceptually different models were tested. This is because relations and information from natural text can be described and extracted in different ways. Each model presented several advantages and disadvantages, discussed in this section. In the end, while they ended up not being used in this work, all of these models are valid and allow for their own specific use cases.

4.1.1 Simple Mentions Model

The **Simple Mentions Model** is the most basic of the described models. The concepts behind it are 1) that relationships between entities might not need to be explicitly captured despite their complexity; and 2) that relations (direct or indirect) happen through articles. Such designs have been used in the context of web archives before, albeit with different goals [18, 34].

In practice, what this model does is connect articles to the entities they mention, while also storing the article's published date as a node, which represents the temporal component of the graph. A simplified representation can be found in Figure 4.1.

The pipeline necessary to generate graphs adhering to this model is also conceptually simple - all that is necessary is Named Entity Extraction (where linking and disambiguation are ultimately optional). The articles are fed into the pipeline, which extracts the entities and associates them to the document.

Due to its design, relationships in this model are explicitly composed of entities and the articles in which those entities are mentioned. Temporal conditions are applied to

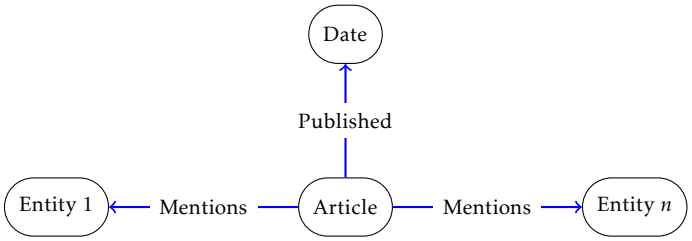


Figure 4.1: Visual description of the Simple Mentions Model.

relations by restricting the publishing dates of articles that the path can traverse.

Advantages The model and pipeline are simple enough to implement with virtually any language (NEE is a common NLP task even in low-resource languages) and run fast regardless of scale. In addition, it allows the user fine-tuned control over what articles to search for.

Disadvantages Because of its simplicity, the information that one can extract from the resulting graph itself is quite limited. For example, performing temporal analysis over it becomes cumbersome, because the temporal aspect is introduced in a very blunt and simplified manner.

4.1.2 Event-Based Model

The conceptual idea behind the **Event-Based Model** is that news most often refer to real world events - things that are happening or have happened - and as such, extracting this sort of information from them seems most natural. When events are extracted, a *type* of event is usually detected, followed by its actors, which are the entities involved. Thus, entities could be related to each other in regards to how they are connected to events. Such an approach has been used in literature, with the goal of facilitating semantic search [22, 56].

By using the Simple Event Model [24] as a basis, such a model could be easily built, where events are associated to actors, which in turn play different roles within the event. In addition, these events would be associated with the article from which they are extracted, and tagged with its publishing date.

The pipeline necessary to build such graphs can easily become complex [22], but the necessary basics start with NEE, followed by Event Extraction or Semantic Role Labelling. When possible, event coreference resolution should also be applied, which identifies when two articles are referring to the same event. The results of such extractions would remain associated to their respective documents.

Similarly to the previous model, entities are not directly connected - relations and paths between them occur through participation in events, which provides semantics to how they might be related together.

Advantages Because news articles are usually focused on events, event-based extraction is an intuitive way to obtain and structure knowledge. In addition, this sort of extraction provides plenty of semantic information in regards to the involved entities, events and how they all relate to one another. In fact, this kind of information extraction is ideal for answering the 5Ws and 1H, which facilitates searching for news articles in a journalistic setting.

Disadvantages Automatically generating graphs adhering to this model is a difficult task. Event Extraction and Semantic Role Labelling are complicated NLP tasks that are not robustly solved in many languages - in Portuguese, there exist only a handful of tools. In addition, these extractors are often fine grained in the events they detect, meaning that for a single text, they return a large amount of events. In a setting dealing with hundreds of thousands of documents, this can quickly scale to an uncontrollable size. This was verified in the initial qualitative evaluation of our experimental graph, but can also be deduced from existing works such as [56] where 51 documents (with an average of 29.4 sentences per text) generated a massive 6178 event entries.

4.1.3 Flat Relations Model

While the previous models did not directly connect entities to each other, the **Flat Relations Model** does just that. The conceptual thought behind it is rather straightforward: the goal of the graph is to relate entities and find connections, thus, we focus on extracting and modeling those connections explicitly. Other works have also developed models surrounding direct relations between entities [19].

Explicitly modeling relations between entities in the context of a graph is simple - there are just edges between the nodes of entities. In addition, the model's relations point to their sources (the article from which they were extracted), are timestamped with that article's publishing date, and can hold any other relevant information, such as a label for the relation or a weight, which depends on the pipeline being used. As such, different tools can generate slightly different models, depending on the capabilities of the extractors.

The pipeline necessary to generate graphs with this model will be composed of at least two components: a NEE module that extracts entities from the text, and a relation extractor that identifies relations within documents and captures relevant information regarding it. Each relation discovered within a document is added to the graph (unless it already exists).

Because relations are explicitly modeled as connections between entities, finding them is intuitive - all that is necessary is to follow the paths, which are always composed solely of entities. Temporal constraints can be directly applied to the paths because the relations are timestamped.

Advantages Modeling extracted relations directly as edges is the most intuitive way to structure them in a graph, which makes understanding and using the resulting database much easier. In addition, it allows the specific analysis of relations over time in a web archive without having to perform workarounds to handle the dates. The pipeline can be simple if desired, which allows it to scale to the levels of a web archive.

Disadvantages While the pipeline itself scales well due to its simplicity, because *every* discovered relation is added to the graph, it can easily grow to large proportions, making it harder to store, use and maintain (similarly to what happened in the **Event-Based Model**). As a result of this large amount of edges between entities, finding non-direct paths also becomes cumbersome and slow, as there are hundreds of possibilities available. Regarding the extraction process itself, relation extraction is a complicated NLP task, and there is a distinct lack of tools for it in languages like Portuguese.

4.2 Condensed Relation Model

Ultimately, the chosen model was an alternate design of the **Flat Relations Model**, which aimed to fix some of its issues, while taking advantage of its simple pipeline and intuitive relation modeling. Conceptually, its basis is that at such a large scale, individual relations from texts are of little importance, and a broader view of how two entities are related within a time span provides more relevant information. It was aptly called the **Condensed Relation Model**. It also combines some aspects of the **Simple Mentions Model**, that allow it to be used in the exact same tasks, while providing extra information on how entities themselves are related.

4.2.1 Underlying Ontology

The model of a graph defines its use and internal meaning - someone who knows why and how a graph is structured can easily make use of it. However, this structure by itself does not provide the graph with any actual semantics. For example, a machine cannot understand it or the data stored within.

These semantics are defined by an *ontology*, which allows data to be shared and understood without having to be explained. RDF Graphs are often associated with one or more ontologies, which provide its nodes and edges with semantics. These ontologies define restrictions for properties and relations - for example, the predicate *presidentOf* must have a *Person* as a subject and *Country* as an object. These restrictions provide the graph with data integrity (by making sure everything makes sense) and allow for inference. Because ontologies provide semantics to data, they also make it that much easier to share - so long as one has the ontology, they can make sense of the data.

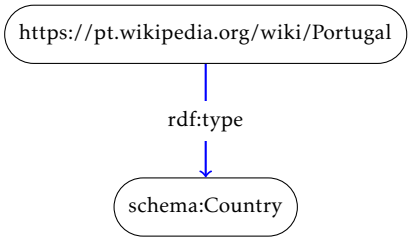


Figure 4.2: Example of a subgraph focusing on the properties of entities.

To the best of our knowledge, fully mapping the Condensed Relation Model, without breaking any restrictions and rules, to an existing ontology is not possible. Creating ontologies is also not a trivial matter and often requires the aid of subject experts [9]. For this reason, a concrete ontology was not developed for the Condensed Relation Model. Instead, we provide generic predicates that can be easily integrated into existing ontologies. These predicates are associated with the URI `<https://ArquiWiz.pt>` and thus prefixed with *arquiwiz*.

In regards to entity types (the classes that define the type of an entity through *rdf:type*), the developed graphs use classes from Schema.org (prefix *schema*). Schema.org is a widely used schema, to which existing knowledge bases, such as Wikidata, either link or have equivalences. This is not ultimately not mandatory in the use of this model, as different systems might prefer different ontologies for typing.

4.2.2 Model Definition

As mentioned previously, the goal of this model is to explicitly connect related entities within a certain timespan (such as years) in order to provide relevant information in regards to how the two are related in that time interval.

Entities are defined by URI nodes and typed (with the property *type* from default rdf) with their respective real world classes in Schema.org. For example, Portugal might appear as the subgraph shown in Figure 4.2. Throughout these examples, the `URI`s for entities are those of Wikipedia, however in reality these can any type of `URI`s.

These entities are connected to each other through relations, however, because these relations have data associated to them (valid timespan or source articles for example) they cannot be simply designed as edges. Instead, a blank node is used to represent the relation itself, which is connected to both entities through the *related* property. The validity of the relation is represented through the *validStart* and *validEnd* properties, while a relation is connected to its sources with the *source* predicate. Any additional property may be added to this relation as well, such as *name* or *weight*. A note to keep in mind is that real-world relations are often directional and asymmetrical, which is reflected in this model. Continuing the example of Portugal, its relation with Marcelo Rebelo de Sousa could be described as in Figure 4.3.

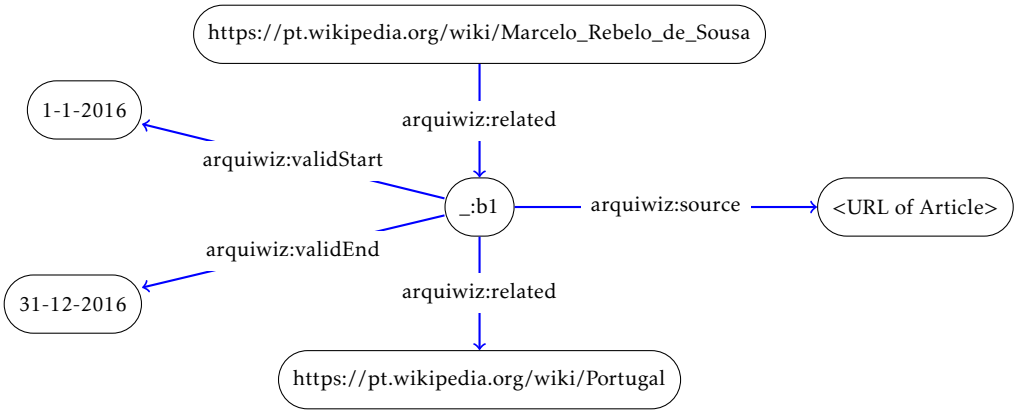


Figure 4.3: Example of a subgraph focusing on the properties of relations.

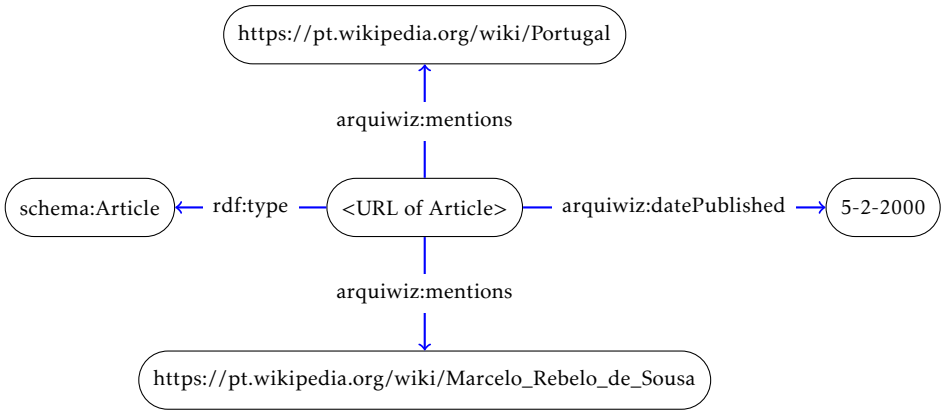


Figure 4.4: Example of a subgraph focusing on the properties associated with an article.

The articles in the graph, from which relations are sourced, also have some important properties. They are typed with the Schema.org *Article* class, have the property *datePublished* to indicate when they were published, and are related to every entity they mention through the *mentions* property. Figure 4.4 shows an example of the properties surrounding articles.

The model is built around entity nodes and blank nodes representing relations. This allows for direct interaction with entities without having to take into account articles during the process. Considering one of the goals of the graph is to also aid in the finding of articles, it makes sense that these appear in the database, also serving for accountability of the existing relations.

Temporal Granularity This model definition is flexible enough to support different types of time intervals - years, trimesters, months or even arbitrary ones. The selected granularity has a large impact on the end result of the graph. Smaller intervals will result in more triples, with less articles per relation - which will likely affect their semantics and accuracy. While there is no restriction to maintain the same interval across time - for example, one might be tempted to use a larger interval in a particular timespan where there are fewer articles - to keep the graph and the relations as homogeneous as possible,

a fixed granularity is recommended. Due to the large span of time covered by the web archive and its news, for the generated graphs, years were used as the time granularity, which means relations were associated with a specific year.

4.2.3 Relations and Path Finding

In the **Condensed Relation Model**, articles do not have to be taken into account when finding relations or paths in this graph, which is an advantage when compared to other models. Relations are more intuitive to use as they are essentially direct. This way of modeling relations leads to certain properties that affect them and the graph as a whole.

Graph versatility. Relations between entities are not *completely* direct because there is a blank node between them. This blank node serves to provide the relation with extra information, such as their validity or anything the designer of the database may want to add. These additional properties are what defines the semantics of that relation. They can be a weight calculated in some way that defines how related the entities are, a label that provides a more specific name, or even embeddings to provide semantics. In reality, by condensing relations and allowing for any property to be associated to them, the model can be used for essentially any manner of modeling relations. In the case of our graphs, due to the existing methods of relation extraction, every relation is assigned a **weight** from 0 to 1 that defines how related its entities are (this relatedness does not differentiate between amicable and unfriendly relations).

Homogeneous structure for relations. Relations in this graph, direct or indirect, will always take the form of **Entity** → **Blank** → **Entity**, repeated one or more times. A *direct* relation is one where the pattern is repeated only once, and thus, the relation was extracted from at least one document. An *indirect* relation is composed of several direct ones. This predictable format allows for the design of query templates that can be automatically adapted to take other entities or filters into account. This homogeneity could also be achieved in other models (such as the event-based one) that boast different types of relations between entities. This would have to be done through inference however, that would in turn require the use of specific ontologies and cause an increase in graph triples.

Constraints. Applying temporal constraints to these relations involves using their validity interval to determine whether to use them or not. There is no need to consider the associated articles or any other node. Applying filters to custom data is just as easy, as this data is at the same level as the validity. For example, one might assign a *types* property to relations (for example, *isPresident*) and filter them directly through that.

Controlled growth Because relations have a controllable temporal granularity, the number of existing relations between entities is controllable, even when there are hundreds of thousands of documents. This in turn reduces the number of paths that can exist between

any two entities, which makes finding indirect relations between them computationally easier.

4.2.4 Overview

The **Condensed Relation Model** attempts to explicitly and intuitively model relations between entities by *condensing* information within specified time spans into single relations, which are stamped with their respective temporal information and more.

The time intervals in which data is condensed can be customized, and serve to create a manageable yet relevant graph in the context of such a large amount of documents. It connects entities directly, allowing for an intuitive use of the graph as well as a predictable style of relations and queries. The temporal aspect is elegantly embedded in the model, allowing for constraints to be effortlessly added to questions.

However, relation extraction is not a common NLP task, making it difficult to implement the pipelines that generate graphs adhering to this model. This is an issue that was mitigated as much as possible with the use of simpler relation extraction processes in our implementations, which are described in Chapter 5.

This model is versatile enough to support differently structured relations between entities, but due to relation-extraction problems, in this thesis, the focus will be on nameless, weighted relations (which is often referred to as the **relatedness** between two entities). This means that all the relations will also have a *arquiwiz:weight* property associated with them.

4.3 Answering questions

The condensed relation model is meant to allow the discovery and study of relations across time. This sort of research involves exploring the graph's relations through various means, which are likely dependent on what exactly is needed. In addition to allowing for this exploration, the graph also has the goal of attempting to answer three types of questions. These questions are based on relations between entities across time, a pattern that our model, and graphs in general, can handle more efficiently than other keyword-based methods. This is because the relations behind the questions are explicitly modeled (directly present in the database) and easily traversed, unlike in other data models.

1. Given two entities (A and B), how strongly are these two entities related, within a certain timespan?

2. Given two entities, a weight and a starting point of their relation (as a year), until when did that relation with that strength continue?

3. Similarly to the previous question, if given two entities, a weight and an ending, when did that relation begin?

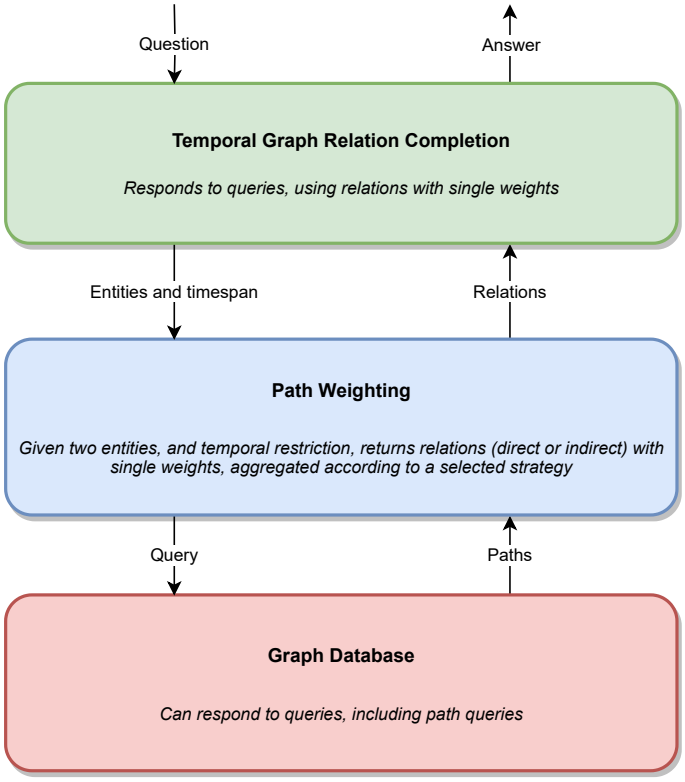


Figure 4.5: System stack of processes and flow of data of when answering questions.

The last two questions can be condensed into a single one - given one of the extremes of a temporal interval, what is the other one?

The graph by itself is not capable of answering these questions - though one might argue that it could respond to the first query by finding a path within that timespan, but simply finding a path is different than providing a single weight for a relation.

As such, we hypothesize that there are two steps to replying to these questions. The first step defines how paths between entities are condensed into single weights, for consequent use in calculations. The second uses the definitions provided by the first one to execute more complex algorithms and respond to questions. An image describing the general stack of processes in answering these questions can be found in Figure 4.5.

4.3.1 Path weighting

A graph specifying relations between entities allows us to naturally model the relation between A and B as a path. This path is composed of a number of *arquiwiz:related* edges where the first edge has A as a subject (starts at A) and the last has B as an object (ends at B). An indirect relation can be qualified as a sequential series of direct relations, that together form a path. A relation, indirect or direct, represents how two entities are related and can be simplified into a path of length ≥ 1.

Because paths are of arbitrary length, the only time a single weight is naturally provided by the graph is when that path is composed by *one* relation - it is direct. However,

indirect relations should be comparable to direct ones, as they are in the real world. As such, there is a need to calculate the weight of a path - of any path - into a single value, which can be compared regardless of whether it is direct or indirect.

The way a path is modeled into a single weight is an important definition that might have drastic effects on the results provided by the system. We introduce two distinct approaches to calculating the weight of a path regardless of its length. Consider a path a set of pairs, $h_i = (w_i, d_i)$ where each pair represents an edge in the path, w_i is the weight of that hop while d_i is how far that edge is from the initial point, starting at 1. For example, the path $\{(0.5, 1), (0.4, 2)\}$ has two steps, the first with a weight of 0.5 and the second with 0.4.

1. **Flat average**: In this case, the total weight of a path ($w(p)$) is the average of the weights of its edges, where $n = |p|$:

$$w(p) = \frac{1}{n} \sum_{i=1}^{n} w_i$$

2. **Weighted Average**: The previous metric does not account for the distance (the number of hops away) of a weight in the path, which in reality might be an important factor. In this scoring approach, we compute a weighted average, in which the weight of a step in the average is its distance to the end of the path:

$$w(p) = \frac{\sum_{i=1}^{n} w_i \cdot (n - d_i + 1)}{\sum_{i=1}^{n} (n - d_i + 1)}$$

These two approaches represent only a portion of different ways to calculate the resulting weight of a relation. One could use the inverse distance, or take into account the types of entities (for example, countries are more commonly mentioned in news and thus relations with them have less of an impact on the resulting weight), or even train a model to perform this step. None of these approaches particularly stands out as more correct.

4.3.2 Temporal Graph Relation Completion

Now that paths can be condensed into single weights - relations between entities can be compared regardless of distance - they can be used to calculate actual responses to the aforementioned questions. Similarly to path weighing, how these answers are computed can vary, and pointing out one option as the best is impossible outright.

For the two types of questions, we introduce multiple methods of computing the answer - each based on a different concept. These methodologies can be combined with any method of calculating the weight of an arbitrary length path and can later be evaluated, to understand which combination is best.

All these algorithms start by finding the **best shortest** paths with temporal constraints. Here, the **shortest** means the distance between the entities - the amount of relations - is the least possible. This choice is made for computational reasons; finding every possible path in graphs of this size would be unfeasible. The used DBMS, GraphDB, when asked for the shortest paths, actually finds the shortest paths along with the paths that have one more edge than those. This works well because it provides us with more options without completely overloading the database. The **best** paths are the paths with the greatest weight, according to the selected strategy. It is also worth pointing out that when a relation between A and B is requested, due to the directional nature of relations, both possibilities (A $\rightarrow$ B and B $\rightarrow$ A) are considered.

The algorithms provided also assume the granularity of years but can be applied to a graph of any granularity, so long as it is consistent across the graph. For example, to answer the same questions in a graph with relations condensed by month, the exact same could be done, but instead of years, months would have to be considered.

Aggregation Strategies To answer the first question - given two entities and a timespan, how related are they - we need to aggregate relational knowledge between the two entities across time and reach a conclusion. To do so, we developed four distinct approaches, each one focusing on a different characteristic of temporal relations and with varying degrees of complexity.

Relations between entities are complex and differ from context to context. As such, even if some methodologies obtain better scores, each one is based on a different concept - a concept that might apply to one relation but not another.

1. **Best@1** and **Best@10**: These methods are based on the concept that relations are defined by their most impactful moments. As such, these methods find the best shortest paths (1 and 10 respectively) in which the composing direct relations of those paths are valid within the given timespan. Returning more possible paths increases the chances of correctly defining the relation and provides users with more references to verify. This method is simple but allows the user quick access to important relations between the two entities and their respective documents; it is also an intuitive and easy-to-understand concept.

2. **TopAverage**: The *Best* methods focus on the most impactful interactions. However, such moments can be exceptions or coincidences. Another reasonable approach would be to not to consider the most impactful connections, but take a more general view and consider the best paths *together*. As such, to calculate the predicted weight, this method uses the k best shortest paths and averages their weights together. Theoretically, if this concept is correct, the larger the k the more accurate the prediction. This methodology is not much more complicated than the *Best* ones and provides the user with a similar amount of documents to reference - the weight, however, may not be as direct as before. In our experiments, k was defined as 10.

3. **AcrossYears** - The previous methodologies took time into account in a loose way; they searched for paths in which relations were valid - regardless of whether the dates of relations *within* a path were associated to each other or not. Some relations might have a deep history and be developed over time, particularly with countries and companies. As such, this methodology focuses on taking time into account more explicitly and aggregating relations across their whole temporal span. To do so, it finds the best shortest path between the two entities *within a single year*, and does so for each year of the timespan. In the end, the weights of these paths are averaged together and represent the weight of the relation. While this method should, theoretically, be more accurate for longer-duration relations, it also requires several times more queries than the previous ones, which in a large graph might result in long waiting times.

An important thing to note in all these processes is that there is a lot of *averaging*. In particular, the fourth and third methods average the weights of relations in a path (according to either the **Flat** or **Weighted** methods) to calculate the result and then average those averages. This type of calculation can oftentimes become biased and that is something to take into account.

Temporal Limit Predicting Where responding to the previous question relied only on analyzing the exiting relations and their evolution over time, predicting the missing end of a temporal interval is much more complex. While we know when the relation stared or ended, and its weight, due to the complex nature of relations, predicting the other temporal limit is not a straightforward task.

To answer this question, we developed two approaches, each with a different concept behind them.

1. **ExpandingWindow** - Similarly to the *Best* methodologies used to predict the weight of a relation, this methodology is based on the concept that a relation is defined by its impactful moments - its times of most interaction. As such, it will try to find the largest temporal span in which the best shortest path is equal to the given weight. To do so, the graph is queried as many times as there are years between the given extreme (inclusive) and the end of the temporal scope - for example, if the start of a relation is 2016, and the temporal scope of the graph ends at 2020, there will be five queries made (2016, 2017, 2018, 2019, 2020). For each query, the span of time in which relations are valid is increased. The other extreme of the best shortest path with the largest interval and the correct weight is considered the right year. If it fails to find a specific year, no result is provided.

2. **ScopedAveraged** - Some relations are not defined by their moments of most interaction but by their overall relatedness over time. As such, to calculate the missing end of a timespan, this approach tries to find the largest time interval in which the

average of weights of the best shortest paths *of each year* equals the provided weight. As such, the graph is queried as many times as there are years between the given extreme (inclusive) and the end of the temporal scope - similarly to before. Each query is associated to a single year, in which the returned path must have all its relations be valid. The calculated extreme is the largest year (or smallest, in the case of trying to find the start of a relation), for which the average of all the weights between it and the known limit is equal to the provided weight.

Similarly to the **Best** methods in weight prediction, the **ExpandingWindow** is simple and prone to misses due to spikes of interaction that do not represent the relation as a whole. It is the simplest result to analyze, however - one must simply look at the relation returned.

ScopedAverage offers a more comprehensive view of the relation over time and thus might theoretically allow for more accuracy - unless the relation itself is episodic, in which case the spikes of interaction are drowned out by the remaining years when they should be defining the relation. It also generates a harder-to-understand reasoning, as there are more relations to analyze.

None of these two approaches, in theory, seems to outshine the other and thus, they were both used during development and evaluation.

5

Natural Language Pipeline

Regardless of the adopted model or context, graphs automatically generated from natural text require a natural language pipeline (often referred to as NLP pipeline) to process them and extract the relevant information. In this section, we present the generic pipeline that has been developed to create graphs adhering to the Condensed Relation Model. This pipeline can be easily adapted to support different extractors and time spans.

5.1 Pipeline Description

Processing pipelines are composed of sequential modules, where the input of a module is the output of the previous one. In this case, the pipeline receives the natural text documents (tagged with their respective metadata) and returns a set of generated graphs adhering to the Condensed Relation Model in RDF. Each module is applied to all documents, before moving on to the next one.

In the context of archives, NLP pipelines should be efficient and run without the risk of crashing, as there are several hundred thousands of documents to process.

The overall pipeline itself, which is visually represented in Figure 5.1, is composed of four main modules, which may then be subdivided into other components:

1. Topical Split. Partitions the whole set of news into smaller groups of pre-defined topics.

2. Entity Extraction. Extracts entities from text and extends the documents with them.

3. Relation Construction. Uses the entities attached in the documents to extract and condense relations according to specific timespans.

4. Data Model Conversion. Uses the extended documents and the relations to construct an RDF graph according to a model.

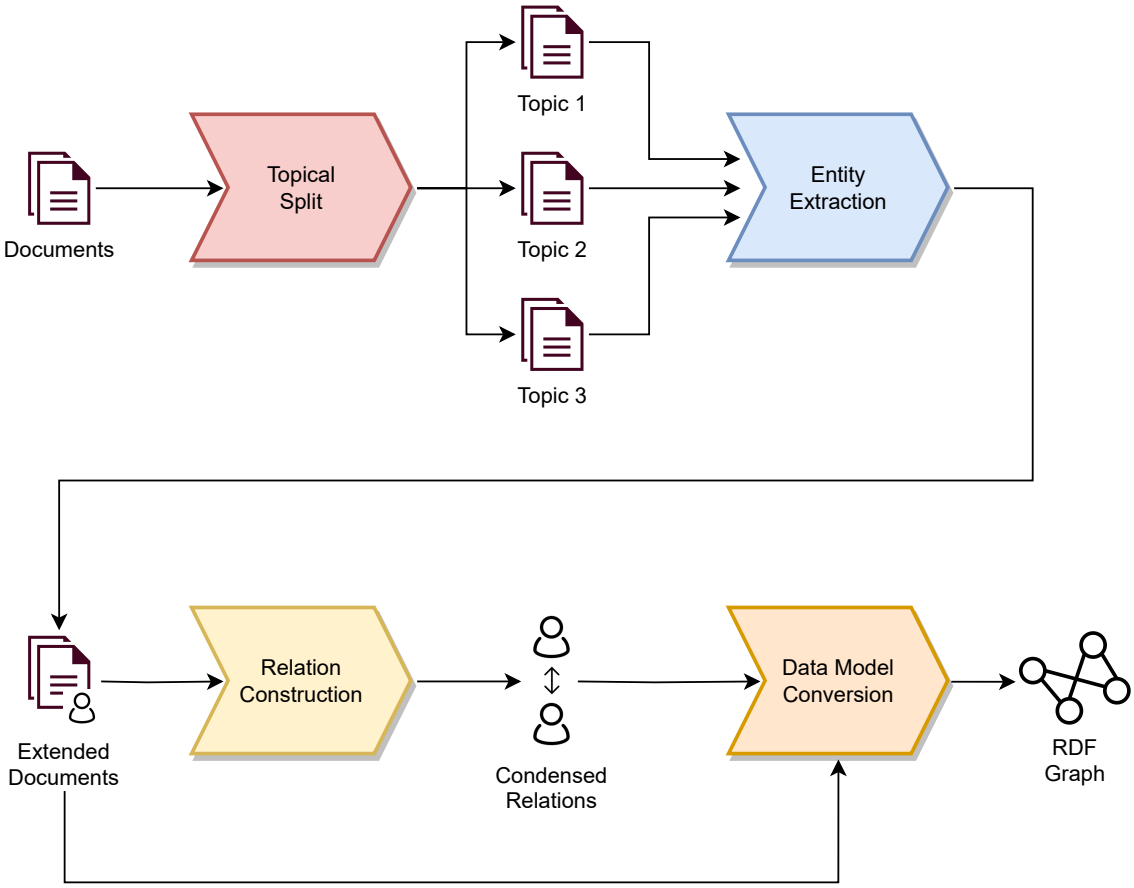


Figure 5.1: Diagram of the natural language pipeline with all four of its modules. The flow of input and output data is represented through the arrows.

5.1.1 Topical Split

The first module of the pipeline is tasked with splitting the document set into pre-defined topics; the rest of the modules are then executed over each subset independently and a graph is generated for each one. This step serves to acknowledge the fact that relations between entities are contextual; for example, it is possible for two countries to be politically divergent, yet have a strong economic connection. This part of the pipeline may be skipped, if such a separation is not desired and instead the rest of the pipeline is run on the whole dataset.

Automatically assigning texts to a topic is an ongoing research problem with several possible methods that vary in complexity and language - for Portuguese, topic classification tools are not yet mature. As such, the problem was approached with a keyword extraction and matching technique. Each topic in the pre-defined set of topics must have an associated list of related keywords - for example, the topic **Environment** can have "climate change" and "carbon emissions" as keywords. The less overlap there is between keywords of different topics, the better.

Given this set of topics and associated keywords, important keywords are extracted

Table 5.1: Keywords for each topic used during the topical split module.

Politics	*acordos políticos legislativas politização liderança reformas fiscais direitos humanos debates governo eletrónico democracia conflitos internacionais parlamento eleições partidos legislação política social relações internacionais política negócios estrangeiros autárquicas programas eleitorais cidadania conflito decisões diplomacia economia governo ministros negociações orçamento presidente da republica republica*
Business and Economy	*negócios economia empresa indústria empreendedorismo investimento mercado finanças banco crescimento económico política económica desenvolvimento sustentável inovação tecnológica emprego comércio economista gestão finanças pessoais comércio internacional empreendedor mercado global estratégia empresarial investimento estrangeiro políticas fiscais setor financeiro crescimento empresarial cadeia de abastecimento concorrência de mercado comércio eletrónico startup economia digital*
Sports	*boxe ginástica futebol basquetebol snowboard karaté futebol internacional automobilismo jogos olímpicos jogador tenis voleibol atletismo surf andebol natação escalada golfe skate hóquei em patins desportista desporto F1 ciclismo esqui triathlon*
Health and Science	*saúde infantil medicina alternativa saúde mental vacinação doenças respiratórias saúde ocupacional envelhecimento saúde pública covid-19 cancro pandemia doenças cardiovasculares diabetes saúde reprodutiva doenças epidemia autismo obesidade saúde bucal saúde sexual sida alimentação saúde do idoso medicamentos epidemiologia inovação pesquisa vacinação avanço doença prevenção saúde medicina ciência tratamento descoberta genética bem-estar farmacêutica*

from the documents in question through YAKE! [7] and compared to the sets of each topic. The topic with the strongest match is selected as that document's topic. This is a simple and lightweight way to split the web pages, that can scale to millions of articles.

In our case, the selected topics were: "Politics" (**Pol**), "Business and Economy" (**B&E**), "Sports" (**Spo**), "Health and Science" (**H&S**). The list of related keywords for each can be found in Table 5.1. Because the topic selection is so strict, many articles were not assigned a topic. To take advantage of all the data, an additional graph (**All**) was also created, which kept all the articles in the same group - it did not assign them topics.

5.1.2 Entity Extraction

This module of the pipeline has the goal of extracting entities mentioned in the texts and disambiguating them - as always this can be accompanied by linking the entity to an external dataset (for example Wikidata), which allows for more information to be retrieved about it if desired. It receives the set of natural documents as well as their respective metadata and returns a set of extended documents. An extended document is built by attaching the entities mentioned in the text to the original document.

The specific entity extraction process is off-loaded from the pipeline as an external function, henceforth called f_{nee}. This function is defined by the developer of the database and is what performs the actual extraction strategy. It can be customized to fit different

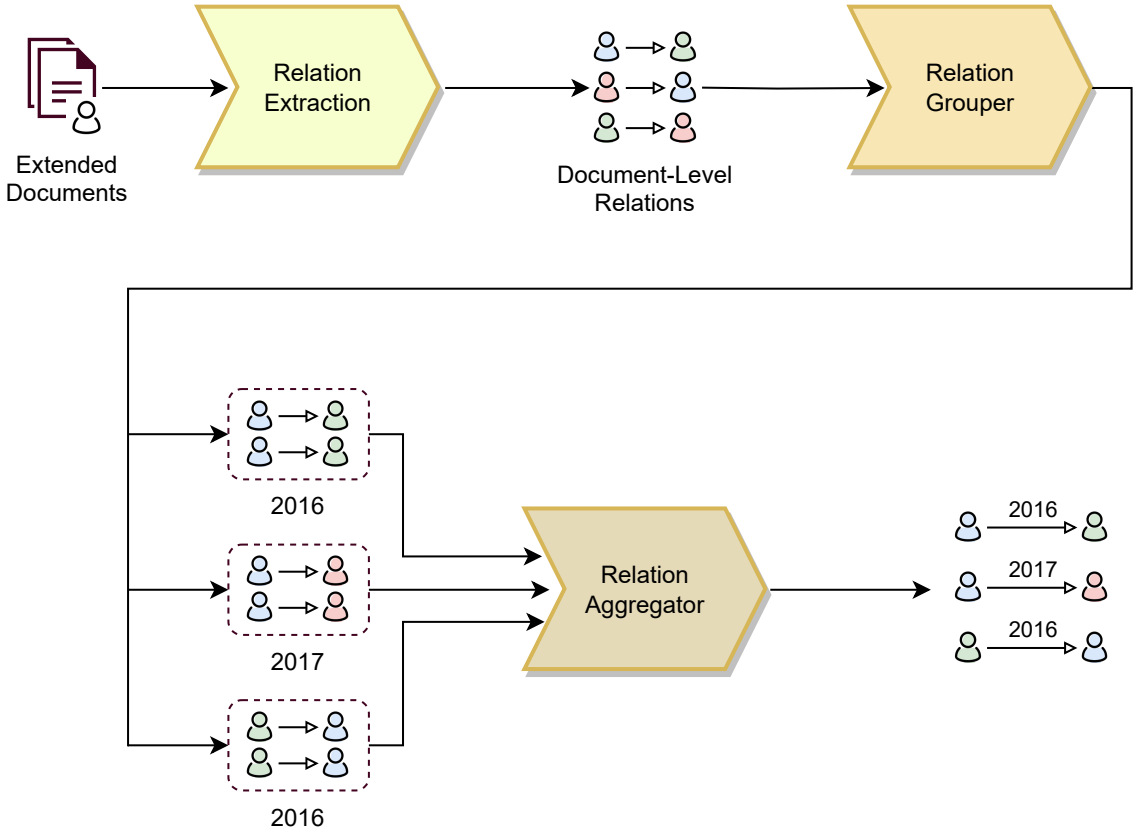


Figure 5.2: Diagram for the data-flow in the relation construction sub-modules.

roles and construct different databases according to their purpose and available tools. For example, a graph that focuses solely on geographic information could have a f_{nee} implementation which extracts only locations.

5.1.3 Relation Construction

The third module receives the set of extended documents and returns the set of directed relations condensed within specific time spans. To do this, the module relies on three sub-modules, each of which can be customized with a function. A visual description of this module can be found in Figure 5.2.

Extractor The first sub-module receives the set of extended documents and for each document, extracts the relations mentioned in the text. It returns the set of all document-level relationships (the set R), which has the relations extracted from all texts. The extracted relations must only involve entities attached to the document and relations have to be temporally tagged, such as with a specific date of occurrence or interval. The temporal interval should also be derived from the document - either its text or metadata. A relation must, at least, have the same components as the triple r.

$r \triangleq (o, d, t)$, where

o and d are the origin and destination entities respectively,

t is the relation's temporal tag.

The grunt of the work of this module is performed by the function f_{re} which receives a document and returns the set of relations between its entities. This function is customizable and thus the pipeline can adapt to different forms of relation extraction. The function f_{re} can also extract additional data to be used by other submodules such as a contextual weight or a label.

Grouper This part of the construction module serves to create the groups of relations that will later be condensed into single connections, according to a specific timespan (years, trimesters, months, etc.). It receives the set of document-level relations (set R) and returns a set of groups, where each group g in turn has a set of relations. Relations within the same group have the same origin and destination entities and the group should have a validity - to indicate which temporal interval it is going to group by. As such, a group can be represented as the tuple g.

$g \triangleq (h, o, d, [t_s, t_e])$, where

$h \subseteq R$ is the set of relations that compose the group,

o and d are the origin and destination entities respectively,

$[t_s, t_e]$ is the group's temporal span.

The function that performs this selection is the f_{grp} function, which receives the set of all relations starting in a certain entity and returns all the groups of relations to be condensed with that entity as the origin. Similarly to previous functions, f_{grp} should be developed considering the database's end goal. The groups of relations returned by the function do **not** have to be disjoint, which allows for greater flexibility in the next module.

Aggregator The final part of relation construction is where the condensing of relations happens. It takes a set of groups and, for each one, creates a single relation according to a specified strategy. This strategy is defined by the f_{aggr} function, which needs to also maintain a group's temporal validity interval. In our case, where the relations should be weighted, the f_{aggr} function also has the role of assigning a final weight to the relation between two entities. For example, a possible implementation of an f_{aggr} function is by taking the contextual weights of a group and averaging them in the resulting connection. While the relations generated by this function are based on existing relations, they are original - they do not exist within the document dataset.

5.1.4 Data Model Conversion

The data model conversion is the simplest of the modules: it receives the extended documents and the constructed relations and creates a graph with them, adhering to the Condensed Relation Model. The process involves first iterating over each document and adding their URLs and mentioned entities to the graph, following their respective specification (articles are tagged with *schema:Article* and connected to entities with *arquiwiz:mentions*). Then, the same is done for each relation that connects the previously added entities.

5.2 Functions Implementations

As previously mentioned, the functions used in each module are provided at pipeline creation and should depend exactly on the context and goal of the graph. These functions define the use cases of the resulting database and as such should be considered carefully.

In our specific case, we implemented several options, which resulted in graphs with different characteristics. These graphs were later evaluated to understand which implementation is better for our news-focused relation-finding use case.

5.2.1 Named Entity Extraction Function (f_{nee})

The f_{nee} function executed in the Entity Extraction module can be implemented in a variety of ways. In the development of this thesis, it went through several iterations before arriving at two stable and usable ones:

1. **Dataset Linking** - This function extracted entities from the natural text through spaCy's[1] named entity recognition models and the results were linked to an external entity dataset through string matching. That dataset was an altered version of the one used in the Major Minors work [40], which is a collection of entity names, split by their real-world classification (country, person, etc.). As such, while the extracted entities were linked to an external dataset, factual knowledge regarding these entities is limited to their type.

2. **Wikifier** - Wikifier[2] is an online API that performs wikification on texts - it tries to extract entities and concepts that have an associated Wikipedia page. In addition, if that Wikipedia page has an associated Wikidata entry, Wikifier will also annotate the entity with the identifier of that entry. This means Wikifier identifies, disambiguates, and links entities in texts to Wikidata. Because it supports the Portuguese language, it is the ideal tool to process our texts. Despite this, Wikifier provides entities with types from Wikidata and not Schema.org, an issue that had to be fixed

[1] https://spacy.io/
[2] https://www.wikifier.org/

manually and is described in Chapter 6. Wikifier is also an online service, accessible through an HTTP API, which incurs some level of delay in the processing of texts.

5.2.2 Relation Extraction Function (f_{re})

The f_{re} function focuses on extracting relations from text. Relation Extraction is a complex NLP task and as such, there is a distinct lack of tools in Portuguese capable of performing it. In this work, the applied methodologies of extracting relations do not strictly adhere to the definition of relation extraction provided in Chapter 2.

1. **Paragraph Co-occurrence** - The simplest form of relation detection breaks the text down into paragraphs and then checks which entities are mentioned together in each one. If two entities *co-occur* in the same paragraph, they are considered to have a relation. This relation has no additional information - it is a pure double (Entity 1, Entity 2). The temporal tag of the relation is the article's publish date.

2. **Sentence Co-occurrence with Embedding Similarity** - In an effort to provide more information regarding the relation *within* the text, a technique using semantic models was developed. First, sentences are encoded into semantic embeddings; then the cosine similarity between entities in the same sentence is used to obtain a weighted relation (the weight is the similarity within the text). This is particularly useful to acknowledge strong relations that might not appear often or vice-versa.

   In our case, for encoding the sentence into embeddings, a Transformer model[3] was used, that creates an embedding for each token. Then, the embeddings of entities mentioned in a sentence are calculated by averaging the embeddings of their composing tokens. This process is visually demonstrated in Figure 5.3. In this case, sentence co-occurrence was used to account for transformers' limited input size. The temporal tag of the relation is the article's publish date.

5.2.3 Relation Grouping Function (f_{grp})

The f_{grp} function has the goal of selecting the time intervals to later condense into single connections. Considering the large number of articles and temporal interval, we adopted a per-year grouping. This means that relations of the same year between two entities are grouped together. This reduces the number of edges to a maximum of 29 (1991-2020) between any two entities, however, in practice, the number of relations will be lower than that; this is because the number of documents below 2015 is small and realistically, despite some exceptions, two entities being mentioned together so many years in a row is unlikely.

[3] https://huggingface.co/intfloat/multilingual-e5-large

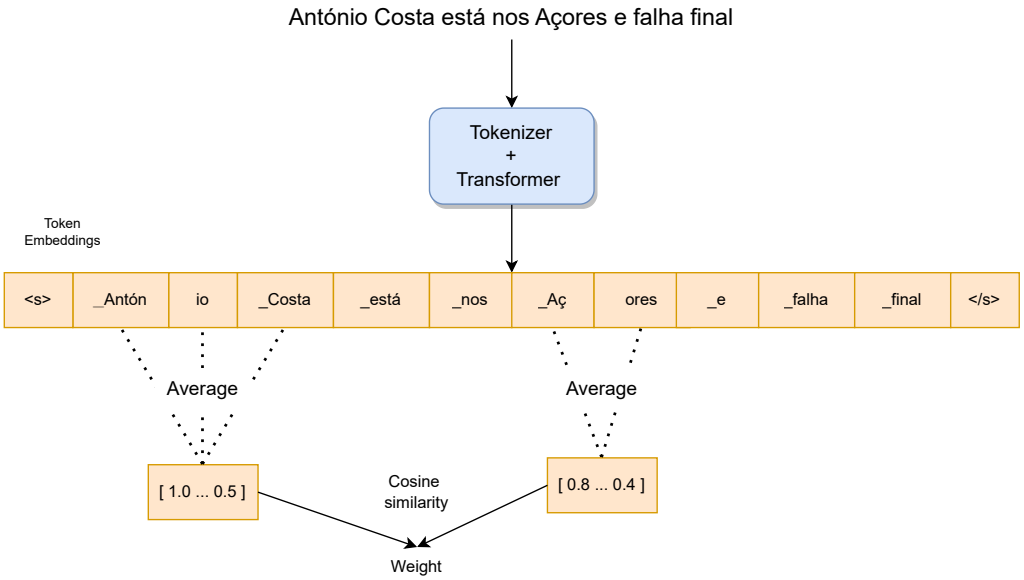


Figure 5.3: The sentence is first passed through a tokenizer which breaks text into tokens, which are then fed into the Transformer model which generates embeddings for each token. Tokens may not exactly match full words. To generate the embeddings of an entity, the embeddings of its composing tokens are averaged and the weight of the relation is the cosine similarity between them.

5.2.4 Relation Aggregation Function (f_{aggr})

The relation aggregation function is one of the most important in the pipeline. Its definition provides the relations in the graph with their final weights. As such, it should be designed and implemented with the end goal of the graph in mind. Because it relies on how relations are extracted - the information it can use depends on the information obtained from those - it should also consider the f_{re} function being used. Realistically, it is likely that each relation extraction function has a matching aggregation function, but this does not have to be the case. The developed proposed variations of this function were:

1. **Statistical Aggregation** - This aggregation function is one that requires no data to be associated with the extracted relations and provides a score based on the *amount* of relations in general. The function is implemented as such, where g is the group to be condensed.

$$f_{\text{stat}}(g) \triangleq \frac{\mid g.h \mid}{\mid \{ r \in R \mid r.o = g.o \wedge g.t_s \geq r.t \leq g.t_e \} \mid}$$

   Colloquially, this means the weight of a relation from entity e_1 to e_2 in a certain timespan is calculated as the ratio between the number of documents that mention that relation in that timespan and the number of documents that mention relations originating in e_1 in that timespan.

2. **Weight Average Aggregation** - When relations are extracted with document-level weights, for example representing how closely related the entities are within the context of a news piece, aggregating them together can follow a different route. In those cases, one can skip looking at the overall picture of relations surrounding an entity and focus solely on performing calculations with those weights. In our case, we decided to assign the condensed relation of a group a weight equal to the average of all the relation weights in that group.

5.2.5 Temporal Aspect

Relations are temporal by definition - they change over time, new relations appear and old ones break off. When modeling and extracting relations between entities, it is thus a crucial aspect to keep in mind.

Ideally, the selected relation extractor would detect a relation's associated temporal expressions in the text. To the best of our knowledge, however, such extractors for Portuguese are not yet available at full capacity.

Even without a specific extractor, however, there exist several different ways to extract and process time from text and documents [6]. For example, it would be possible to extract temporal expressions from text, normalize them, and use them to tag the relations detected in the same paragraph or sentence.

This technique might seem feasible at first glance and is similar to ones that have been used before, for example by Yeung et al. [82] where topics extracted from news were associated to temporal expressions mentioned in their sentences. It is a viable technique and with time could be further studied to understand its impact. However it brings up its own host of issues in the context of relations, because there is no direct way to understand whether a temporal expression is referring to the relation in the text or something else entirely. For example, take the following sentence, obtained from an archived news article[4]:

Na sessão de abertura interveio também o presidente da Câmara de Lisboa, António Costa, que assegurou que a cidade está hoje mais preparada para responder a uma catástrofe deste tipo do que estava há 25 anos.

(The Mayor of Lisbon also spoke at the opening session, António Costa, who assured that the city is today more prepared to respond to a catastrophe of this type than it was 25 years ago.)

This sentence mentions two entities, Antonio Costa and Lisbon (or Lisbon City Hall depending on the extractor), and a temporal expression, "25 years ago". Yet, this expression has no connection at all to the relation between the entities. Just using a simple

[4] https://arquivo.pt/noFrame/replay/20191026000111/https://www.dn.pt/portugal/sul/incendio-do-chiado-mudou-forma-de-combate-aos-fogos-3320116.html

method - such as applying a temporal expression to all the relations mentioned in the same paragraph - without very specific rules is likely to have a large degree of error, in comparison to the current approach.

Temporally tagging relations extracted from text with expressions from the same text is an ongoing research problem that requires complex sub-tasks such as document understanding. Because of this, applying it in this context cannot be done in such a straightforward manner.

Instead, due to a lack of better tools, the generated graphs use an article's publish date to tag the relations from it. This is a predictable format that is also easy to trace - the user *always* knows the source of that tag, even without reading the article - but might generate some incorrect extractions, as sometimes news refer to past, or even future, events and relations. By grouping relations in years, this inaccuracy is, theoretically, somewhat minimized, as a news article is often no more than a few months late (or in anticipation) to explaining something.

Time as an Entity To more explicitly take time into account, one might suggest using time expressions in the text as entities instead. As such, they would be related to people, organizations, countries, etc. just as any other entity is.

While this could be a way of capturing events - dates that are often mentioned are likely to be the date of something important - it would be more likely that the graph became polluted by a large number of nodes without much meaning at all. Not only would this bloat already large graphs, but in the case of statistical weights (weights that depend on the frequency of relations), these meaningless nodes would influence the overall weight of more truthy relations.

5.2.6 Dealing with scale

The implementation of this pipeline presented several challenges throughout its development. Besides the lack of mature tools for relation extraction or entity disambiguation/linking, the largest problem with implementing a pipeline such as this is the scale of the data. There needs to be a balance between stability and speed.

Due to the scale, processes had to be linearized - data is read from disk, processed, and then written back to disk - to provide more stability and avoid running out of memory. For some functions, this was possible without much issue. The entity extractor, the topic splitting mechanism, and data model conversion are all parts of the pipeline that focus on one particular element at a time and as such can read, process, and write that element without difficulty. RDF conversion was initially done through the RDFLib library, but due to its memory-based implementation which often crashed, it was switched to a self-made solution.

However, this type of methodology cannot be directly applied to relation construction, because a more global view of the relations is necessary, which is particularly obvious in

the **Statistical Aggregation** variation. To account for this, we use an external database, in this case, Redis[5], that stores the relations so these can be accessed quickly without overloading the process' memory.

The pipeline is thus stable regardless of scale, while maintaining a respectable temporal efficiency, even if it was not optimized to its fullest potential - it runs mostly sequentially. The slowest part of the pipeline as a whole was the **Wikifier** function, which took several days to complete (even parallelized) due to its online nature and the size of the dataset.

5.3 Resulting Graphs

The function implementations applied to the pipeline had a total of three combinations used, shown in Table 5.2. For each combination, five graphs were generated, four for the selected topics and a fifth one with no assigned topics (due to the loss of a large portion of unassigned documents). Statistics regarding each of these graphs can be found in Table 5.3 (with a bar chart representing the triples of each graph in Figure 5.4).

Table 5.2: Combinations of function used in the pipeline - grouping function was the same for each one.

Name	$f_{\mathbf{grp}}$	$f_{\mathbf{re}}$	$f_{\mathbf{aggr}}$
LPS	Dataset Linking	Paragraph Co-ocurrence	Statistical Aggregation
WPS	Wikifier	Paragraph Co-ocurrence	Statistical Aggregation
WSA	Wikifier	Sentence Co-ocurrence with Embedding Similarity	Weight Average Aggregation

The most obvious difference between the LPS (**L**inked **P**aragraph **S**tatiscal) graph and the other two is its smaller, yet still substantial, size. This happens because the extraction module uses direct string matching, which causes it to miss entity mentions and thus less information is extracted. The number of triples and relations also varies vastly between WPS (**W**ikifier **P**aragraph **S**tatistical) and WSA (**W**ikifier **S**entence **A**verage) - this is due to the scope of the co-occurrence. Because WPS uses a paragraph scope, more entities are captured at once than if only sentences were used, and thus more relations are extracted. It also becomes clear that the number of entities and mentions in the graph depends solely on the f_{nee} function, as WSP and WSA use the same implementation and possess equal statistics in those fields.

It is worth noting, however, that quantity does not equal quality. While there might be an increase in accuracy in some methodologies that combine weights of different paths (such as **ExpandingWindow** for temporal prediction), there are techniques (such

[5] https://github.com/redis/redis

Table 5.3: Numerical statistics of the generated graphs.

Graph	Topic	Triples	Entities	Articles	Relations	Mentions
WPS	Pol	26.06M	21.18k	100.94k	5.38M	2.3M
	B&E	10.73M	14.0k	37.03k	2.58M	0.8M
	Spo	4.03M	11.77k	14.38k	1.08M	0.24M
	H&S	10.18M	11.95k	29.2k	2.33M	0.78M
	All	218.9M	96.45k	984.12k	41.96M	19.67M
WSA	Pol	10.95M	21.18k	100.94k	1.92M	2.3M
	B&E	4.5M	14.0k	37.03k	0.91M	0.8M
	Spo	1.46M	11.77k	14.38k	0.32M	0.24M
	H&S	4.03M	11.95k	29.2k	0.77M	0.78M
	All	91.07M	96.45k	984.12k	14.83M	19.67M
LPS	Pol	2.41M	2.38k	101.28k	0.29M	0.42M
	B&E	0.74M	1.97k	37.11k	0.12M	0.11M
	Spo	0.29M	1.53k	14.41k	0.05M	0.04M
	H&S	0.55M	1.52k	29.27k	0.08M	0.09M
	All	14.9M	5.84k	984.12k	1.23M	2.98M

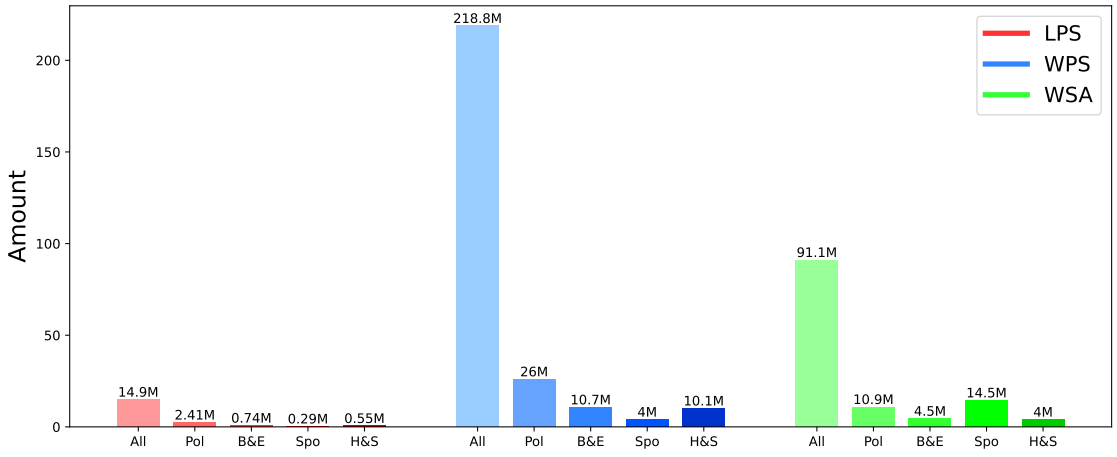


Figure 5.4: Number of triples per graph.

as the **Best** methodology for finding weights) that do not look at multiple paths when performing the final calculation. More paths means more chance of getting the correct one, but also of capturing more trash. As such, just from their size it is not possible to know whether one graph models the world more correctly then another - to assess that, more appropriate methods (such as the ones discussed in Chapter 7) are required.

6

Incorporating Existing Knowledge

The goal of the graph is to model relations between entities over time, as extracted from news articles. However, news articles often omit common knowledge information or facts. As such, combining our graph with existing knowledge bases will allow for more complex queries and possibly more accurate relations.

6.1 Sources and Incorporation

Wikidata [73] is a crowd-sourced effort to take wikipedia's knowledge and place it in a more structured format. Because it is crowd-sourced and not automatic, it is very accurate in its construction. Due to its solid facts, several works use it for tasks such as entity linking or machine learning.

One of the named entity extraction functions provided in Chapter 5 uses the Wikifier tool. This tool detects Wikipedia entities in the provided text, and even provides the appropriate Wikidata identifiers for the extracted entities. As such, by using this tool, it is possible to acquire additional information regarding the detected entities, from Wikidata.

We hypothesize that there are three main ways to include this additional knowledge into the generated graph, which differ in nature and complexity.

1. **Entity typing**
2. **Additional Facts**
3. **Weight Shifting**

6.1.1 Entity Typing

As the name states, entity typing has the goal of providing entities with their respective types. According to the Consensed Relation Model, it is highly recommended for these annotations to exist in the graph.

In Wikidata, and most knowledge bases, it is common for entities to have more than one type, which then have different types connected as subclasses of each other. This

is a consequence of Wikidata's complex class system. For example, Portugal is both a country, a sovereign state, a colonial power (temporally tagged as truthful between 1415 and 1975), and an OECD[1] country. In turn, a country is a subclass of a territory.

Another consequence of Wikidata's complex class system appears in equivalences to Schema.org. Not all of Wikidata's classes have an equivalent Schema.org type, which means that it is possible for some entities to be left without a type. For example, in Wikidata, Samsung is typed as a chaebol, which is a specific type of south Korean conglomerate. Cheabol is in turn a subclass of a conglomerate, which is a subclass of corporate group which is a subclass of organization. Only *organization* here has an equivalence to Schema.org. As such, it was decided that the entities in the graph would be typed with the equivalent classes of their direct types along with those of their super classes.

When Wikifier extracts entities, it is also capable of providing their types. However, these types are strictly direct types and can be any class from Wikidata. Instead, to acquire the types of the entities in our graph, we used Wikidata's public SPARQL endpoint and performed the query in Listing 2 for each entity in the graphs. The statistics shown in Chapter 5 already take these extra triples into account.

```
SELECT DISTINCT ?type
WHERE {
    wd:ID wdt:P31/wdt:P279* ?wikitype .
    ?wikitype wdt:P1709 ?type .
    FILTER(regex(str(?type), "https://schema.org"))
}
```

Listing 2: SPARQL query to fetch Schema.org types of an entity. The predicates wdt:P refer to properties in Wikidata, where P31 is "instance of", P279 is "subclass of"and P1709 is "equivalent class". ID is replaced with the id of the current entity.

6.1.2 Additional Facts

News are an invaluable source of information for the common reader and even machines. However, their focus lies on descriptions of current happenings, not on factual knowledge. As such, they often leave out general information that a reader is likely to know already such as who the current president is. This in turn affects the way text is written, and as such, the information that can be reliably extracted from these artifacts.

The developed system as it is may be enough for analysts - it allows them to explore data and answer questions. However, if existing factual knowledge is required, such as which countries belong to the EU, the users must search for that information themselves, and then explicitly incorporate the entities that satisfy their conditions into the query.

This can be cumbersome when there are a lot of entities to consider, or it is a complex query that requires facts from several different sources. It is also necessary to manually

[1] Organisation for Economic Co-operation and Development

update the query whenever there are changes, which becomes a problem when it is embedded in a larger application.

Ideally, the query itself would access this factual knowledge, thus avoiding the explicit definition of statements and entities, and guaranteeing the query's functionality even if data changed. For this to be possible, the graph would have to include common knowledge facts.

One of RDF's main focuses is data exchange and as such, combining knowledge was made to be as simple as concatenating two sets of triples together (so long as there are no domain-name clashes). A graph linked to Wikidata (its entity nodes point to Wikidata entities) can easily add statements from the external knowledge base - all they have to do is access the triples and add them to the graph. Other data models (such as property graphs) might have required complex processing to convert between the different representations.

Adding Wikidata triples to the generated graphs thus allows for more complex queries that take existing *facts* into account. For example, with the appropriate triples having been added, the following question could be answered.

During their election year, to which political parties other than their own, were the presidents of Portugal strongly related?

This query involves three factual statements that are not naturally present in the graph. To answer such a query, the graph would need knowledge regarding 1) who were the presidents of Portugal, 2) when they were elected and 3) which political party they belonged to. The SPARQL query that corresponds to this question is in Listing 3.

This query was executed on the WPS **Pol** graph - to which the necessary triples were previously added. Only Marcelo Rebelo de Sousa had any matches, with the strongest political party he was related to being the Socialist Party with a weight of 0.516.

By including external triples, we increase the expressiveness and flexibility of the graph, allowing for even more complex studies of the social environment. Adding triples needs to be done with care, however, as external knowledge bases are often massive in size - on the scale of billions of triples.

6.1.3 Weight Shifting

Another way to incorporate existing knowledge into the graph is by using factual statements to alter the weights of the relations. To do so, we took advantage of Knowledge Graph Embeddings.

The KGE set used was one learned from Wikidata5m [75], a subset of Wikidata with 5 million entities, using the RotatE [69] method, downloaded from the Graphvite library[2]. This KGE does not take time into account and while there are methods for calculating Temporal Knowledge Graph Embeddings there is a current lack of proper datasets from

[2] https://graphvite.io/docs/latest/pretrained_model

```
PREFIX p: <http://www.wikidata.org/prop/>
 PREFIX pq: <http://www.wikidata.org/prop/qualifier/>
 PREFIX wdt: <http://www.wikidata.org/prop/direct/>
 PREFIX wd: <http://www.wikidata.org/entity/>
 PREFIX ps: <http://www.wikidata.org/prop/statement/>
 PREFIX schema: <http://www.schema.org/>
 PREFIX arquiwiz: <https://ArquiWiz.pt/>
 PREFIX rdf: <http://www.w3.org/1999/02/22-rdf-syntax-ns#>

 SELECT ?president ?other_party ?weight
 WHERE {
     ?president p:P39 ?statement.
     ?statement ps:P39 wd:Q322459;
                pq:P580 ?start_date.
     ?president wdt:P102 ?party .

     ?president arquiwiz:related ?b .
     ?b arquiwiz:related ?other_party .
     ?b arquiwiz:weight ?weight .
     ?b arquiwiz:validEnd ?tag .
     ?other_party rdf:type schema:PoliticalParty .

     FILTER(?other_party != ?party
                 && YEAR(?tag) = YEAR(?start_date))
 }
 ORDER BY  DESC(?weight)
```

Listing 3: SPARQL query to find the political parties to which the presidents of Portugal were very strongly related (looking in both directions) on their year of election, other than their own. This query uses properties such as "held office" (P39) and "start time" (P580) along with the "President of Portugal" entity (Q322459).

Wikidata that consider time. The most widely used one is Wikidata12k [15] which contains only 12 thousand entities, none of which are Portugal, for example. When applying this methodology, the only relations affected by weight shifting were the ones in which both entities existed in the KGE dataset. As such, weight shifting does not alter the amount of triples in a graph.

This methodology is generic to any graph which provides its relations with weights. The concept is that the similarity between two entities in a set of knowledge graph embeddings can represent the weight of the relation between those entities. As such, by combining two sources of different knowledge - the one extracted from news and the one extracted from a structured base - we can achieve a higher correctness.

To apply this method, one must iterate over every relation in the graph, take its weight, find the similarity between those two entities in the KGE set, and apply a function, which replaces that relation's weight. In this case, the applied function was a weighted average between the two:

Table 6.1: Extended combinations of functions used in the pipeline - grouping function was the same for each one.

Name	$f_{\textbf{nee}}$	$f_{\textbf{re}}$	$f_{\textbf{aggr}}$	**Shifting**
LPS	Dataset Linking	Paragraph Co-ocurrence	Statistical Aggregation	No
WPS	Wikifier	Paragraph Co-ocurrence	Statistical Aggregation	No
WSA	Wikifier	Sentence Co-ocurrence with Embedding Similarity	Weight Average Aggregation	No
WSA^{wiki}	Wikifier	Sentence Co-ocurrence with Embedding Similarity	Weight Average Aggregation	Yes

$$0.8 \times W_{e_1 \to e_2} + 0.2 \times \text{sim}(\text{KGE}, e_1, e_2)$$

The KGE similarity is considered less important in the final value due to it lacking the temporal aspect of the relation. The previously calculated weight ($W_{e_1 \to e_2}$) has a superior influence due to the goal of the graph - to model relations *across time* in news. This methodology was applied over the **All** component of the WSA graph, generating a new graph with the exact same statistics. The updated list of all graphs can be found in Table 6.1.

There are a multitude of other ways to incorporate existing knowledge into a graph through weight shifting. For example, Ostendorff et al. [48] combine, through the use of a learned layer, contextual sentence embeddings with KGE before measuring the similarity between them. A similar approach could be applied to the **Sentence Co-occurence with Embedding Similarity** relation extractor - provided a training dataset existed to train the layer.

Other options might stray from KGE and adopt a more rule-based approach and direct use of Wikidata. For example, relations of a certain kind would boost or lower the weight of an edge, if they existed in the knowledge base.

7

Evaluation and Results

A graph is as useful as it is correct and expressive. Because the graphs being discussed are automatically generated, a degree of error is expected. However, that degree must be assessed, to ensure that it is above a certain usable threshold. In this section, we present a methodology developed to assess the quality of the generated graphs and the system capabilities of fulfilling the desired applications.

7.1 Methodology

For evaluating the graph, we decided to forgo extensive manual evaluation of the extracted data or resulting triples for determining the quality of the knowledge base like other works [8, 22]. Instead, we use an automatic method, which also allows us to assess the quality of the system as a whole and not just the graph database. To do so, we had to design a *gold standard* dataset (or ground truth) - since none in this context exists - to which the system (graph, queries and algorithms) was compared.

The selected methodology is similar to the one taken to evaluate the *link prediction* task in Knowledge Graph Embeddings [74]. For each element of the dataset, we hide one of its components and ask the graph to complete it - the evaluation's score is determined by how many correct completions it makes.

7.1.1 Dataset

The gold standard dataset designed contains 128 manually created entries that relate pairs of entities, such as how they would be in the real world, where each entry contains two entities, a weight - which represents the correct strength of the relation - and an associated topic, to which the weighted relation belongs. Entries can also have associated timespans that represent the interval in which that relation with that weight exists. These timespans are optional, meaning there are relations without them that should be considered across the entire temporal axis. A few examples of these entries can be found in Table 7.1. During the evaluation of a topical graph, only entries of that topic are to be used.

Table 7.1: Sample entries from the developed gold standard dataset, which relates entities as they would be in the real world. A blank temporal interval means that the relation is general across time whereas a "-" means that the relation lasts to the system's latest year.

Entity A	Entity B	Weight	Topic	Start	End	Reason
Portugal	Marcelo Rebelo de Sousa	5	Pol	2016	-	Elected as the country's President
António Mexia	EDP	4	B&E	2006	2020	Period as President
Telma Monteiro	Cristiano Ronaldo	2	Spo			World renowned Portuguese athletes

The relations considered between two entities can either be more direct - those people worked together, that organization has their main office in that country, etc. - or more indirect - two people share similar values, even if they never interacted or two countries have similar histories. Of these types of relations, essentially only direct relations have associated time intervals.

To develop these relations a simple process was taken - an entity, related to Portugal, was chosen, and that entity's Wikipedia page was studied. From there, direct relations are easy to extract: any link to another Wikipedia page in the text can essentially be treated as a direct relation. Indirect relations are harder to find - this is where a graph such as the developed ones would have been useful - and focused mostly on trying to search for common points of interest by studying the history of several different entities or by using common knowledge. Ideally, this dataset would be more extensive and constructed by subject experts.

In the graph, the weights assigned to relations are of a continuous nature, between 0 and 1, for maximum flexibility. To simplify manual assignment and later comparison, however, the weights in the ground-truth dataset are assigned as discrete values from 1 to 5 ($\{1,2,3,4,5\}$). A weight from the graph in $[0,1]$ is converted to its equivalent discrete value according to the function f. The discretization of the weight always happens before it is compared to the ground truth.

$$f(x) = \begin{cases} 1 & \text{if } 0 \leq x < 0.2 \\ 2 & \text{if } 0.2 \leq x < 0.4 \\ 3 & \text{if } 0.4 \leq x < 0.6 \\ 4 & \text{if } 0.6 \leq x < 0.8 \\ 5 & \text{if } 0.8 \leq x \leq 1 \end{cases}$$

This dataset was developed to be generic and usable in different situations. As such, it was created without any particular entity extractor in mind, which in the end caused some of its entities to not appear in the graphs. In these situations, when one of the entities

is not present in the graph being evaluated, the entry is skipped, thus not affecting the resulting score.

7.1.2 Evaluation Targets

The developed system is composed of different parts, each of which is going to be evaluated throughout this methodology.

- **Combinations** - Considering the flexibility of the developed pipeline and the different implementations of its functions, the main graph groups will be compared to one another in order to understand whether a particular implementation is more accurate. The groups are WPS, LPS, WSA, and WSA^{wiki}. WPS and WSA use Wikifier as their entity extractor, while LPS uses a fixed dataset. WSA uses embeddings and sentence co-occurrence to extract relations while WPS and LPS use paragraph co-occurrence. WPS and LPS also rely on statistical aggregation to calculate the final weights of their relations while WSA uses the average of the document-level relations. WSA^{wiki} had its weights shifted through KGE from Wikidata.

- **Topics** - The news dataset was split into four topics: Politics (**Pol**), Sports (**Spo**), Business and Economy (**B&E**) and Health and Science (**H&E**). Each topic generated a graph of its own, as well as an extra **All** graph which did not assign any topics to its news and instead used all of them. The results of the different topics will serve to understand if there is an increase (or decrease) in accuracy when the topic of a relation and news article is taken into account.

- **Path Weighting** - The two different methodologies of weighting paths are to be examined, to understand how the two techniques affect overall quality. Under study are the **flat average**, which simply averages the weights of the path, and **weighted average**, which takes distance into account by using it to weigh a relation's effect on the average.

- **Completion Algorithms** - Each of the completion algorithms is going to be analyzed in conjunction with the previous traits, to understand which combination most accurately responds to questions. The existing methodologies for predicting weights are **Best@1** and **Best@10** which return the highest weight paths, **TopAverage** which averages the top ten results, and **AcrossYears**, which calculates a weight for each year and then averages them. For predicting temporal spans we will analyze the **Expanding Window** technique, which increases the span of valid relations until the best path is the correct weight, and **Scoped Average** that calculates the best relations for each year and finds the combination with the weight.

7.2 Tasks

In link prediction [74] any component of the entry can be hidden. In our case, we instead focus on two particular possible tasks: **Weight Masking** and **Temporal Masking**. Each of these tasks in turn matches with one of the questions the system is meant to answer, to test how well the system is capable of fulfilling its designed applications. An additional task, **Graph Density**, was performed, to assess the average distance between entities.

7.2.1 Weight Masking Task

In this task, the weight of the entry is hidden, and the system is used to estimate it, given the timespan (which can be a specific interval or the whole axis). Essentially, we evaluate whether relations that should be in the system, are in fact present, and if weights across time are correct (according to the ground truth triples). Formally, for each entry $(e_1, e_2, w, [t_s - t_e])$, the system attempts to find the weight relation between e_1 and e_2, where that relation exists in the time interval of $t_s - t_e$. If the relation does not exist - when both entities are present in the graph - it is considered a 0 regardless of the metric.

Metrics To quantify this task, the following metrics were used.

- **Accuracy** ($\Delta = 0$ or $\Delta = 1$) is the average of how many correct weights were predicted, in comparison to all the non-skipped entries. Here, a predicted weight ($\hat{w}$) is considered correct, if $| w - \hat{w} | \leq \Delta$. As such, the accuracy serves to show how correct the predictions of weights are in general.

- **F1(j)** is the harmonic mean between a graph's *precision* (P) and *recall* (R). It is often used in machine learning and is particularly useful when the distribution of classes in a dataset is not uniform, which is the case, where the ground truth has a different number of entries for each possible weight. As such, F_1 score is calculated for each possible weight value ($j \in \{1, 2, 3, 4, 5\}$). Precision and recall for a certain weight j are calculated in regards to *true positives* (tp), *false positives* (fp) and *false negatives* (fn).

  In the following notation, w represents the correct weight in the ground truth, $\hat{w}$ is the predicted weight, and j is the value for which F1 is being calculated.

  True positives are predictions that are equal to the value for which the score is being calculated and are correct. As such, they adhere to the predicate $\hat{w} = w \wedge \hat{w} = j$.

  False positives are predicted weights that are equal to the value being calculated - meaning the graph predicts that the weight of that relation is j - but that is not correct (the ground truth differs). Thus, false positives are defined by $\hat{w} = j \wedge \hat{w} \neq w$.

  False negatives are incorrect predictions, in which the ground truth is the target weight. False negatives are predictions that follow $\hat{w} \neq j \wedge w = j$.

$$P(j) = \frac{tp}{tp + fp} \qquad R(j) = \frac{tp}{tp + fn} \qquad F_1(j) = \frac{2 \times P(j) \times R(j)}{P(j) + R(j)}$$

7.2.2 Temporal Masking Task

With the goal of evaluating how well the graph predicts the time intervals of relations, this task masks one of the timespan's components and asks the graph to complete it. Each entry is run twice - once with the end of the interval masked, and another with the start. More formally, for each entry $(e_1, e_2, w, [t_s - t_e])$, the system tries to complete the temporal interval of that relation, by predicting t_s and t_e one at a time. Similarly to the previous task if the graph is unable to provide a concrete year or the relation does not exist, that entry is considered 0.

Metrics To quantify this task there is only one metric, **Accuracy** ($\Delta = 0$ or $\Delta = 1$), which is the average of how many correct years are calculated in relation to how many entries were used. Like with weights, accuracy measures how correct the predictions of a year are. Each entry is separate, even though entries are considered twice - once for each extreme of the timespan. A predicted year is considered correct if it differs no more than Δ years from the ground-truth value.

7.2.3 Graph Density

The six degrees of separation concept (also known as the six handshakes rule) theorizes that any two people are no more than six social connections away. Similarly, the six degrees of Kevin Bacon is a game to test whether the social distance to Kevin Bacon (called the Bacon number) is also always six or less.

To understand how densely connected entities in the generated graphs are, two experiments were conducted, each based on one of the previous concepts. These experiments measure the average length of several hundred paths as well as the distribution of the different lengths. These experiments were both performed on the **All** graphs of all combinations.

1. The first experiment (the **Handshakes Experiment**) has the goal of figuring out what the average distance between any two entities is. To achieve this, we randomly sample 800 pairs of distinct entities and measure the paths between them.

2. The second (the **Mariza Experiment**) serves to understand the average distance to a *specific* entity. In this case, the fadista Mariza serves as a central point. She was selected because she is well known and Portuguese, but belongs to a very specific context, thus guaranteeing that she is not often mentioned in articles of just any kind. To achieve this, 800 entities are randomly sampled and their distance to Mariza is measured.

Table 7.2: Best accuracy results for the All graphs of each category, including the weighted and algorithm combination to obtain them.

Graph	Weighting	Method	Accuracy $\Delta = 0$	Accuracy $\Delta = 1$
WPS (**All**)	Flat Average	Best@10	0.492	0.792
LPS (**All**)	Flat Average	Best@10	0.506	0.775
WSA (**All**)	Weighted Average	Best@10	0.365	0.765
WSA^{wiki} (**All**)	Weighted Average	Best@10	0.322	0.739

7.3 Results and Discussion

The evaluation of the system provided results for the different tasks, which further allowed us to reach some conclusions in regard to both the graphs and the overlaying techniques. The results of every metric for the Weight and Temporal Masking Tasks can be found in Annex II.

7.3.1 Weight Masking Task

Predicting the weights of a relation given its composing entities and temporal span provides us with a lot of data and is the easiest of the two tasks, while also arguably being the most used application of the system.

Best Graph When dealing with the prediction of weights, an initial question one might ask is which graph is the best. Best here is a loose definition, but can be associated with higher accuracy.

To answer this question, Table 7.2 is presented, which indicates for each of the **All** graph variants, the best score and the combination of factors that obtained it. From analyzing this table, several patterns can be noticed.

1. The best method in all of them was Best@10, which likely characterizes the nature of most relations as episodic. It also shows that relations by themselves are already correct - aggregating different relations together or across time does not necessarily provide better results.

2. Graphs using embedding-based contextual weights (WSA and WSA^{wiki}) seemed to favor weighted averages for the calculation of their relation weights, while statistical aggregation (WPS and LPS) benefit more from using flat average.

3. The results themselves are rather positive, with the best of the best graphs achieving relatively high accuracy. Even the more strict accuracy of $\Delta = 0$, although unsurprisingly not as elevated as $\Delta = 1$, provides accurate scores around half the time in statistical graphs.

Table 7.3: Comparison of different weight calculation algorithms across WSA and WPS.

Graph	Weighting	Method	Accuracy $\Delta = 0$	Accuracy $\Delta = 1$
WPS (**All**)	Flat Average	Best@1	0.283	0.717
		Best@10	0.492	0.792
		Average	0.275	0.717
		Across Years	0.15	0.35
WSA (**All**)	Weighted Average	Best@1	0.278	0.643
		Best@10	0.365	0.765
		Average	0.278	0.617
		Across Years	0.287	0.739

4. Shifting the weights of WSA through KGEs from Wikidata (resulting in WSA^{wiki}), seemed to *lower* the accuracy of a graph as opposed to the expected increase. This might be due to its lack of temporal aspect or simply an incompatibility of contexts.

While this table allows for some conclusions, it raises questions that require further study of the results: such as how other algorithms compare to the supposed best, Best@10, and how the flat average actually differs from the weighted average in the WSA graphs.

Comparing prediction algorithms To compare the different algorithms in two different settings, Table 7.3 presents the results of each algorithm in the **All** graphs of WPS and WSA, which raise several observations:

1. Best@10's higher accuracy is closely followed by other algorithms, despite being the most flexible one conceptually.

2. Best@1's accuracy in $\Delta = 1$ is quite elevated, showing that oftentimes - though not all the time - the best path, the one with the highest weight, ends up being accurate. Surprisingly, it is also better than taking the overall average of the top results. This once again confirms the concept that relations are often defined by episodic moments.

3. While performing a per-year aggregation in statistical graphs does not seem to bode good results - AcrossYears method has the lowest accuracy in WPS - the same cannot be said in embedding-based graphs, in which taking time into account increases the overall score. As such, combined with the previous results, one might conclude that weight calculations in embedding-based graphs are more susceptible to distance and time than in statistical graphs, which seem to favor the weights of relations already present in the system.

Table 7.4: Comparison of weighting methods within the WSA graph.

Graph	Weighting	Method	Accuracy $\Delta = 0$	Accuracy $\Delta = 1$
WSA (**All**)	Flat Average	Best@1	0.278	0.617
		Best@10	0.278	0.617
		Average	0.278	0.617
		Across Years	0.278	0.617
WSA (**All**)	Weighted Average	Best@1	0.278	0.643
		Best@10	0.365	0.765
		Average	0.278	0.617
		Across Years	0.287	0.739

Comparing Weighting Techniques The best graphs displayed a difference between embedding-based and statistical ones when it came to their weighting technique. To understand how these weighting techniques, flat and weighted averages, compare within the same graph, Table 7.4 is presented. It shows the several graph question-answering algorithms with both weighting techniques, in the WSA (**All**) graph.

The first thing that becomes immediately noticeable in these results is that every method has the exact same accuracy in flat average. In fact, by looking at the F_1 values of these functions, shown in Table II.1, one will notice that for weights 1 through 4 this metric is zeroed out, with $F_1(5) = 0.435$. $F_1 = 0$ means that precision or recall is zero (this makes the calculation of F_1 impossible and thus 0). Precision and recall can only be zero if the system failed to guess a single correct value of that weight. This means that the WSA graph only correctly guessed ground truth tuples with a weight of 5 and its precision and recall are decently balanced.

These values are explained when the resulting successful (both entities exist) weights returned by the system are studied - as shown in Figure 7.1 - and it becomes clear that WSA's guesses were essentially *all* 5, regardless of the metric used. This shows that WSA is highly biased towards high scores and thus perhaps a different threshold should be

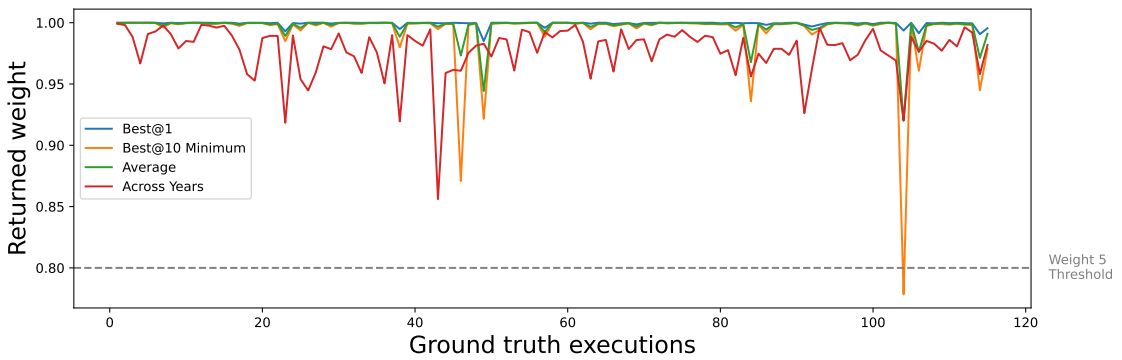


Figure 7.1: Distribution of results across ground truth tuples and methods. Best@10 Minimum represents the lowest of the 10 values analyzed by the algorithm.

Table 7.5: Evaluation results of LPS graphs, using the Flat Average weighting method.

Weighting	**Method**	**Graph**	Accuracy $\Delta = 0$	Accuracy $\Delta = 1$	**Entries**
Flat Average	Best@10	All	0.506	0.775	89
		Politics	0.708	0.875	24
		Sports	0.471	0.882	21
		Health and Science	0.4291	0.81	22
		Business and Economics	0.727	0.864	17

Table 7.6: F_1 scores of the WPS graphs, calculated with flat average and Best@10 methods.

Weighting	**Method**	**Graph**	$F_1(1)$	$F_1(2)$	$F_1(3)$	$F_1(4)$	$F_1(5)$
Flat Average	Best@10	All	0.095	0.24	0.323	0.453	0.366
		Politics	0.333	0	0.357	0.417	0.444
		Sports	0	0.25	0.308	0.364	0.364
		Health and Science	0	0.222	0.118	0.424	0.56
		Business and Economics	0	0	0.32	0.32	0.375

used for the several weights. It also explains the increase in accuracy when distance and time were more explicitly taken into account, as this served to smooth out the bias.

Topical quality The previously shown results all referred to the **All** components of the graphs. This choice was made due to their larger - and thus more representative of the real world - size. However, due to the contextual nature of relations, the generated topical graphs are also of importance. As such, the different topical graphs of LPS and their accuracy with the combination of Flat Average and Best@10 method are shown in Table 7.5.

Relating to each graph is the amount of entries that actually went into calculating these scores. Most topics were applied in similar numbers with the exception of the lower Business and Economics. This makes it harder to generalize the results between the several topics, but such a comparison is still worth making.

From analyzing these results, it is clear that some topics wield much superior accuracy in $\Delta = 0$, while others share a similar range as the graph without topics. The looser $\Delta = 1$ accuracy experiences a large increase in all topics, which surpass the **All** graph in this metric. This superior quality is particularly noticeable in the **Politics** and **Business and Economics** topics, whose accuracy is considerably greater than **All** in any Δ. **Sports** and **Health and Science** present lower accuracy in the stricter rule ($\Delta = 0$), indicating that these topics might be harder to model and capture than the rest.

These positive results in comparison to the **All** graph might be an indication that taking topics into account when sourcing news will provide more accurate relations. These results are common across graphs, except for the **Sports** topic, which often falls behind the **All** graph, accentuating the difficulty of representing this theme.

Table 7.7: Best accuracy results for the All graphs of each category, including the weighted and algorithm combination to obtain them.

Graph	Weighting	Method	Accuracy $\Delta = 0$	Accuracy $\Delta = 1$
WPS (**All**)	Flat Average	Expanding Window	0.047	0.085
LPS (**All**)	Flat Average	Expanding Window	0.028	0.066
WSA (**All**)	Flat Average	Scoped Average	0.113	0.113
WSA^{wiki} (**All**)	Flat Average	Scoped Average	0.113	0.113

3, 4 and 5 Weight bias In graphs with an associated topic it also becomes clear that higher weights, three through five, have a higher F_1 score in the Best methodologies - with lower weights sometimes being zeroed out. An exception are the WPS and LPS graphs with the Across Years method, which are inverted, having higher values of F_1 in the lower weights and zeros on the highest, likely due to a smoothing out of the high spikes of interaction by years with little to no relation.

To exemplify, the F_1 scores of WPS with the Best@10 method and flat average technique are shown in Table 7.6. This trait indicates that the graph systems are slightly biased towards higher weights, achieving on them a better balance between recall and precision than on the lower ones.

7.3.2 Temporal Masking Task

Predicting the temporal extreme of a relation is a difficult task, even with the provided information. The evaluation of the developed system (graph and algorithms) serves to show this much.

The best results for each **All** graph are shown in Table 7.7, along with the components that lead to their calculation. Right away it becomes obvious the low accuracy of any metric and combination, which shows the difficulty of predicting such a complex range of values. Despite this, it is possible to identify some patterns from the presented results:

1. Statistical graphs still seem to be favoring episodic spikes of interaction and existing values of the graph (by achieving better results in the Expanding Window method) while WSA still benefits more from taking time explicitly into account - likely to smooth out the previously discovered bias.

2. Embedding-based solutions have more accuracy while using the flat average for the discovery of dates - which contrasts with the trend in the previous task where these graphs benefitted more from the weighted average. This is perhaps due to the temporal restriction being more important than the distance during the calculation of each year's path.

Table 7.8: Comparison of temporal prediction between topics in LPS graphs.

		Expanding Window		**Scoped Average**	
Weighting	**Graph**	$\Delta = 0$	$\Delta = 1$	$\Delta = 0$	$\Delta = 1$
Flat Average	All	0.028	0.066	0.009	0.019
	Poltics	0.026	0.079	0.026	0.026
	Sports	0.0	0.0	0.0	0.0
	Health and Science	0.042	0.042	0.0	0.0
	Business and Economics	0.038	0.077	0.0	0.0

While in the previous task topics had a greater accuracy, the same is not true here. This is evidenced by Table 7.8, which shows the results of both algorithms with flat average in the LPS topical graphs. The results are generally similar to the ones on the **All** graph, though some topics completely failed to predict one correct date regardless of the Δ. These same results also show us that the different methods can serve to provide quite different results, with Scoped Average having rather low accuracy in the LPS graphs. Like before, **Sports** seems to be the least correct topic, which is similarly verified in almost all other graphs, with **Sports** often having zero accuracy. Once again serves to show that this topic might be particularly challenging to capture.

Failure Analysis Despite the previous conclusions, these results do not allow for a more detailed description of what might have gone wrong - all that is known is that the accuracy is low, but not by how much or why. The metric of accuracy is too rigid for this, it does not tell us what the degree of the error actually is.

To further analyze the problems behind this, two statistics were studied: the average distance to the correct year and the percentage of non-responses. The first calculates, for each predicted year, the distance to the correct temporal edge and averages these values together. The second shows the percentage of non-skipped entries where the algorithms failed to predict a year. These statistics for the LPS graphs can be found in Table 7.9.

Through these statistics, it becomes clear that the main source of low accuracy is not due only to incorrect year predictions, but mostly because the algorithms are too often unable to even find a response. However, when they do, the prediction is often incorrect - the average distance is much larger than one in any case. By analyzing these results it becomes clear that the degree of error, which is directly proportional to the average distance, is quite large.

Of the two techniques, **Scoped Average** has the highest rate of non-response. This is probably due to its more strict policy of looking only for paths in which all composing relations are of the same year - if a relation has a year with less weight, it will affect the calculation as a whole.

Table 7.9: Statistics for the LPS graphs regarding the average distance between the predicted and the correct year, as well as the percentage of times algorithms did not predict a year.

		Expanding Window		**Scoped Average**	
Weighting	**Graph**	Avg. Distance	Non-response	Avg. Distance	Non-response
Flat Average	All	5.516	65%	5.5	90%
	Pol	4.125	58%	3.25	89%
	Spo	5.4	79%	-	100%
	H&S	3.0	94%	-	100%
	B&E	7.143	71%	-	100%

In cases like this, it is difficult to discern whether this means that the graph's relations are incorrect - the evolution of entities across time is not well represented - or whether the algorithms themselves are not well designed.

If we were to analyze the graph's correctness, to assess the source of low accuracy, the best way would be to select a series of relations and perform a manual evaluation to verify whether their evolution over time makes sense. This manual processing is taxing, however, due to the size of the temporal scope and complexity of real-world relations - a subject expert would be ideal for this task.

Such a process might not be required in this case, as the positive results in regard to the weight prediction tasks indicate that the graph has a decent degree of correctness, and thus the majority of the failure probably comes from the algorithms' designs.

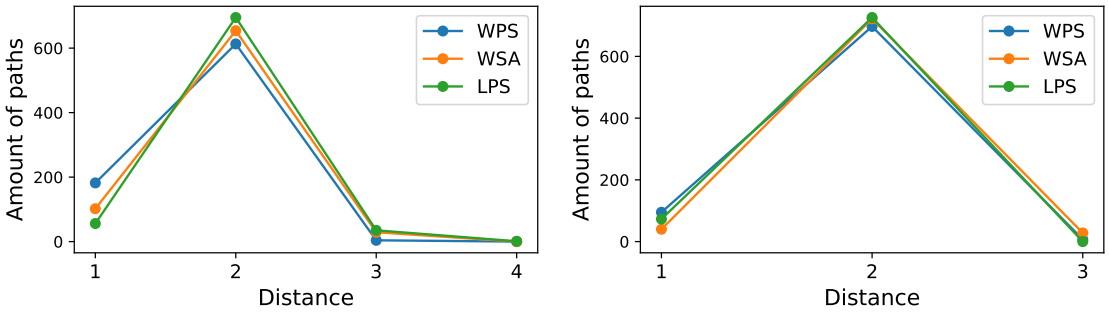
In general, both techniques would require a deep dive with controlled experiments to understand whether they are correct and what exactly to change. For example, by constructing a small graph that is known to be correct and executing the algorithms on it - thus evaluating *only* the algorithms - we would be able to understand if they are conceptually valid and if so, why they have such a degree of error.

A good start to fix them, should the problem be that their methodologies be too strict, would be to loosen their weight comparison. Instead of looking for relations with the *exact* same weight as the given one, they could search for stronger or weaker weights, in case of failure. This way, the percentage of non-responses would decrease, though that does not necessarily mean that the accuracy itself would increase.

There is also the possibility that both algorithms would have to be re-designed, or discarded altogether - meaning the problem does not lie in their details but in the concepts behind them. These problems also go to show just how complicated the design of temporal prediction algorithms is.

Table 7.10: Results for the graph density experiments. *Average length* is the average length of the paths that were found. *No Path* is the number of paths that were not found. The total number of paths that were searched in execution was 800.

	Handshakes Experiment		**Mariza Experiment**	
Graph	Average Length	No Path	Average Length	No Path
WPS	1.78	1	1.89	1
LPS	1.98	13	1.91	1
WSA	1.91	15	1.98	10



(a) Handshakes Experiment.

(b) Mariza Experiment.

Figure 7.2: Path length distribution in graph density experiments.

7.3.3 Graph Density

The results of these experiments serve to provide insight in regards to the overall density - how close in general entities are to each other - of the generated graphs. Table 7.10 holds the average length of the paths returned by the tasks, while the distribution of lengths can be found in Figure 7.2a for the **Handshakes Experiment** and Figure 7.2b for the **Mariza Experiment**. Throughout these experiments, the non-existence of a path was not qualified for the overall average. This is because it cannot be considered a zero, as that would lead to a smaller average distance, which is untrue. Instead, the amount of paths that were not found is displayed in the results table (Table 7.10).

From these results it is possible to conclude that in general, the graph is quite dense - the average density is always less than two in either experiment, with WPS being the most dense of the three. It is also possible to conclude that LPS and WSA are less connected than WPS - there are likely more disconnected (isolated) subgraphs, which leads to the system not finding paths between entities in different patches.

The difference between the results of the two experiments is minor, which means that this average distance is likely common across entities. Some exceptions might be larger-scale entities, such as high-profile organizations or countries. In either case, the majority of the values concentrate on the distance of two relations away, rarely going farther than three.

7.4 Evaluation Overview

The evaluation methodologies and artifacts presented throughout this section serve to provide a way of analyzing large scale entity-relation graph systems in order to understand their correctness and structure. These focus on the specific applications of our use case of predicting relation weights and temporal extremes.

The graphs generated by the developed pipelines presented good results in the prediction of weights, across different methods, with a slight bias for more elevated weights. Through this evaluation, we were able to conclude that shifting weights through KGE seemed to lower accuracy, rather than increase as expected. Separating the graphs in topics did however provide a boost in weight predicting accuracy, particularly in the case of **Politics** and **Business and Economics**, where **Sports** revealed itself to be a harder to capture and represent topic.

The prediction of temporal extremes had subpar results, due to the complexity of the task at hand. In this case, topics did not seem to provide improvement, with **Sports** once again showing itself as the hardest to model of the selected ones. After a failure analysis, it was concluded that the main problem lies in the algorithms often being incapable of providing any predictions. The reason behind these issues was discussed, with possible solutions being presented.

As for the graph's density, through the **Handshakes** and **Mariza** experiments we were able to ascertain that the graphs are quite dense, with most entities being, on average, two relations away from each other.

8

ArquiWIZ: Graph Based Web Archive Exploration

Over the course of this thesis, a prototype was developed to allow users to engage with the created graphs in an appealing manner. This prototype focuses on two particular use cases of the graph - relation finding and analyzing the direct relations of an entity over time.

8.1 Overview and Context

The prototype - by the name of ArquiWIZ - was developed alongside two other theses [2, 36], both of which also focused on the archive. The goal of the prototype was to make the features developed during this thesis as easy to access as possible. As such, there was a large emphasis on making it intuitive and visually appealing. Despite this, because it is not the focus of this thesis, it serves as a proof of concept - a prototype - rather than a full-fledged application.

Two particular features allowed by our system were included in this prototype, each of which is presented as a specific page, accessible through an initial home screen. A particularly interesting aspect of the prototype is the mascot agent - a small wizard that provides some assistance and guides the user through the visualizations. Similarly to evaluation, discretization of the overall weights of relations is performed to allow for more obvious strengths. Weights are converted from $[0,1]$ to *very weak*, *weak*, *medium*, *strong* and *very strong*. Due to its more manageable size, the *LPS* **All** graph was used in the prototype.

In the context of graphs, it is common to imagine a visualization similar to that provided by most graph-data visualizers, such as Neo4J workbench [76], where the database, its nodes and edges, is explicitly drawn exactly as a graph that can be explored. This way is normally natural and intuitive but becomes harder to use when time is included. With this in mind, other projects have chosen more "story-like" types of visualizations [51] where entities are represented as lines that intersect in time. This is usually done in smaller graphs, however, with fewer points of connection.

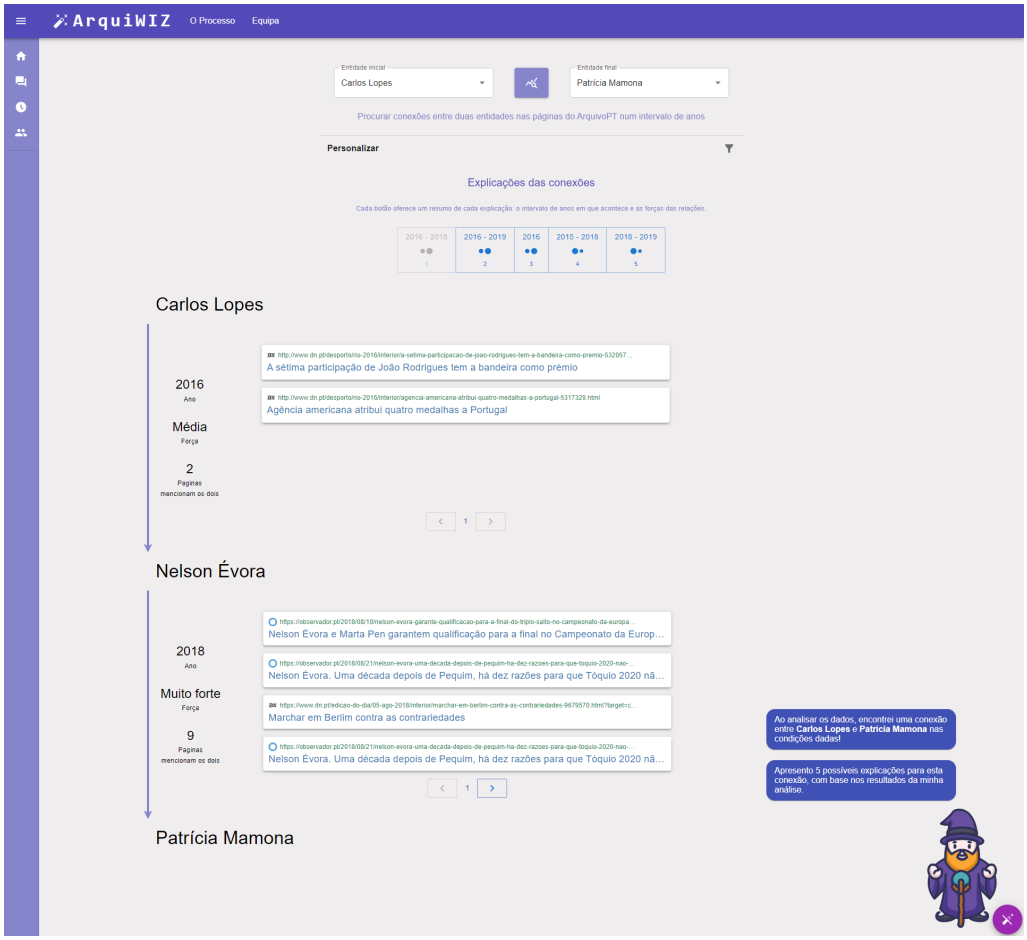


Figure 8.1: Web interface of an example relation found by the relation finding feature. It is possible to explore different paths and the news of each composing relation.

8.2 Relation Finding

The simplest way to use the system - but by no means the least useful - relation finding allows the discovery of a path between two given entities, in an optional time span and with an optional minimum strength. Such a task uses the length of the paths to sort the relations, defaulting to **Best** + **Flat Average** for ties.

The goal is to allow users to find relations, that perhaps they did not know of, along with news mentioning these relations, or their component connections. By providing optional parameters, the user can also tweak the search to their needs. An example of the web interface for this feature can be found in Figure 8.1.

8.3 Environment Analysis

Analyzing the relational environment around an entity is studying how its relations with other entities have changed over time and attempting to find patterns or the lack thereof. To allow for this kind of study, we developed the Bubble View.

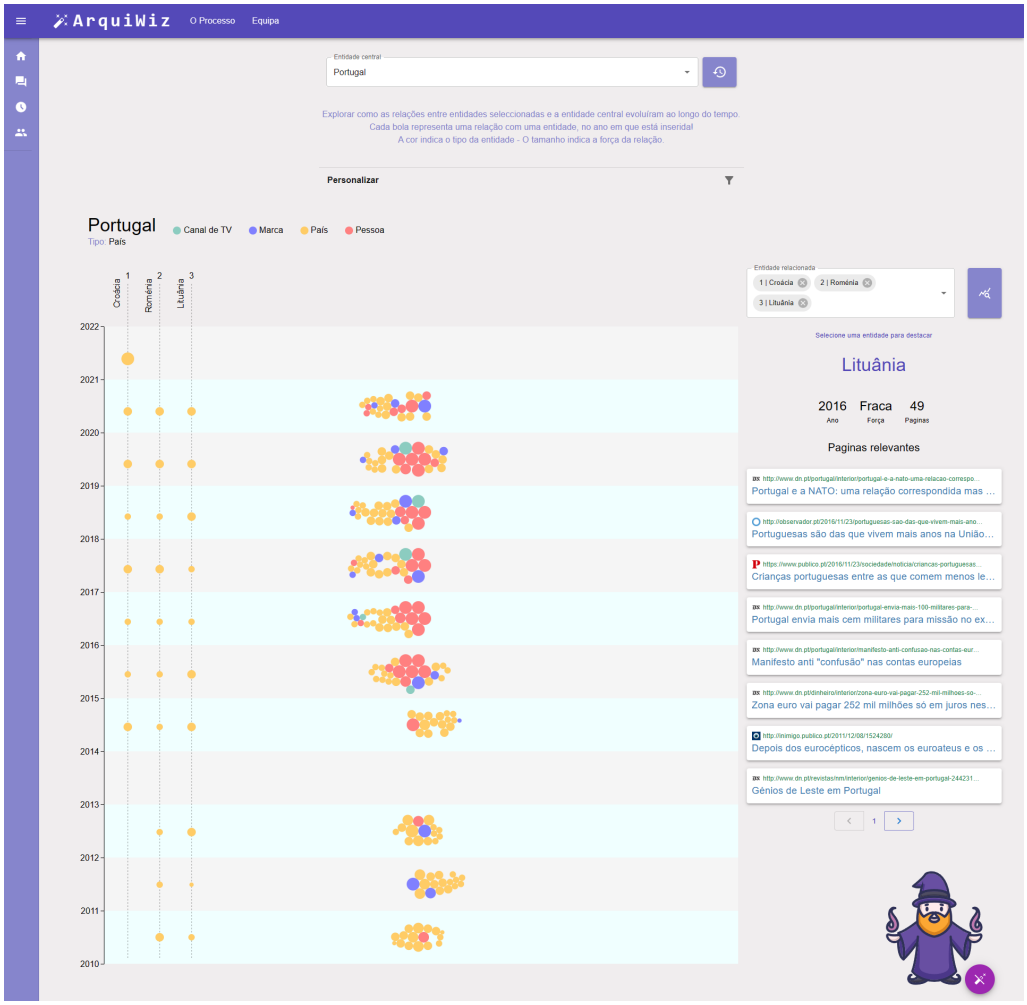


Figure 8.2: Web interface of the bubble view of an entity's relational environment. Entities can be brought into focus or highlighted.

A bubble view is centered around a specific entity that is being studied. This visualization is a series of areas stacked sequentially, where each area represents a year. These areas are filled with bubbles, where each bubble represents a direct relation with that entity, its size represents how strong that relation is, and its color shows the type of the related entity.

The view is interactive where one can hover entities to highlight or select them to bring them into focus. One can also control how many circles are shown per area and what types of entities are actually considered. Relations are selected on a per-year basis - the strongest k relations are selected for each year. Since we are only dealing with direct relations, there is no need to perform any weight calculation. An example of the web interface for this feature can be found in Figure 8.2.

9

Conclusions and Future Work

This thesis was built on the hypothesis that an entity-relation graph could be automatically constructed from news texts, obtained from the ArquivoPT web archive, by developing an NLP pipeline and accompanying graph model.

Throughout its course, we designed the Condensed Relation Model and built a flexible and customizable pipeline to create RDF graphs adhering to it. This pipeline partitions news into topics, extracts entities, and aggregates their relations across specified timespans. Due to the lack of common knowledge in news, we developed different ways to enrich these databases with existing facts, provided by structured knowledge bases. Considering one of the main applications of the system is to answer specific questions, several methods were developed for this end. These methods are divided into a two-step process; firstly, paths are weighted into a single value, which is then used in the algorithmic computation of a relation's weight or date. These methods were evaluated with a novel methodology, accompanied by a manually created ground truth, designed for this particular use case. As a way to simplify access to this system by non-technical users, a web interface was also developed.

By applying this framework of construction over a subset of news from ArquivoPT we developed a total of sixteen different graphs, each built in different ways or with different topics. The largest of this graph contained over 230 million triples and had 982 thousand news articles for its creation.

Besides the developed databases themselves, we deliver several other contributions, such as the framework necessary for creating entity-relation graphs, the overlaying question-answering system, and a sample benchmark and extendable methodology for evaluating them, regardless of domain, language, and origin.

The results of evaluating the graphs were generally positive, with the prediction of the relatedness between two entities achieving accuracy as high as 0.875 (out of 1) in some cases. Temporal prediction proved itself to be much harder, which is evidenced by the lower accuracy of this task, which maxed out at 0.125. This low accuracy was studied alongside additional statistics, such as the average distance to the correct year, in order to understand the root and scale of the this problem - this distance varied between 3

and 7.143 in the studied cases, which represents a large degree of error. Through this evaluation, a small bias towards higher weights was also discovered, being particularly evident in one combination of factors. The density of the resulting graphs - how close entities are to each other - was studied and the average distance between any two entities was found to be less than two.

We consider that we have achieved the end goal of this thesis by developing not only a well-defined structured model but also the necessary practical and theoretical means to apply that model to any set of data and later evaluate it. The generated systems have the potential of aiding journalists and historians alike, by answering relation-based questions and allowing the intuitive study of the relational environment of entities across time.

9.1 Future Work

Despite the initial good results, there is always room for improvement and we list here some aspects which we hope to explore:

- **More complex relation extraction** The methodologies used resorted to simple relation extraction based on co-occurrence. It would be interesting to explore how more complex - and at the time, less stable - ways of extracting relations from the text. Such options might include developing our own model, trying to use textual parsing, or many other avenues in the area of relation extraction.
- **Better temporal algorithms** With time, the development of better temporal prediction algorithms would be one of the priorities, in order to increase the accuracy of the system in this application and task.
- **Dedicated extractor** Similarly to relation extraction, a more specific entity extractor and Wikidata linker would be ideal in this situation. Wikifier is a good tool, but seemingly captures too many entities - a more fine-tuned approach would allow for more control over the growth of the graph.
- **More comprehensive ground truth** The ground truth, while painstaking to make, is in reality short for graphs on the millions of triples scale. With the aid of subject experts and more time, a more comprehensive dataset would be created.

Bibliography

[1] R. Angles et al. "Foundations of Modern Query Languages for Graph Databases". In: *ACM Computing Surveys* 50.5 (Sept. 2018), pp. 1–40. DOI: 10.1145/3104031. URL: https://doi.org/10.1145/3104031 (cit. on pp. 12, 14, 15, 17).

[2] J. Arvana. "Time-aware Question-Answering for the Portuguese Web Archive". MSc diss. NOVA School of Science and Technology, Oct. 2023 (cit. on p. 74).

[3] S. Barbosa, M. Gonçalves, and M. Magalhães. "Padrões linguísticos do femicídio na imprensa escrita portuguesa". In: *Revista da Associação Portuguesa de Linguística* (Nov. 2020), pp. 21–36. DOI: 10.26334/2183-9077/rapln7ano2020a2. URL: https://ojs.apl.pt/index.php/rapl/article/view/87 (cit. on p. 2).

[4] V. Brattka et al. "Measuring the Complexity of Computational Content: From Combinatorial Problems to Analysis (Dagstuhl Seminar 18361)". In: *Dagstuhl Reports* 8.9 (2019). Ed. by V. Brattka et al., pp. 1–28. ISSN: 2192-5283. DOI: 10.4230/DagRep.8.9.1. URL: http://drops.dagstuhl.de/opus/volltexte/2019/10327 (cit. on p. 11).

[5] J. Camacho-Collados and M. T. Pilehvar. "From Word To Sense Embeddings: A Survey on Vector Representations of Meaning". In: *Journal of Artificial Intelligence Research* 63 (Dec. 2018), pp. 743–788. DOI: 10.1613/jair.1.11259. URL: https://doi.org/10.1613/jair.1.11259 (cit. on p. 9).

[6] R. Campos et al. "Survey of Temporal Information Retrieval and Related Applications". In: *ACM Comput. Surv.* 47.2 (Aug. 2014). ISSN: 0360-0300. DOI: 10.1145/2619088. URL: https://doi.org/10.1145/2619088 (cit. on p. 51).

[7] R. Campos et al. "YAKE! Keyword extraction from single documents using multiple local features". In: *Information Sciences* 509 (2020), pp. 257–289. ISSN: 0020-0255. DOI: https://doi.org/10.1016/j.ins.2019.09.013. URL: https://www.sciencedirect.com/science/article/pii/S0020025519308588 (cit. on p. 45).

[8] G. Carnaz, V. B. Nogueira, and M. Antunes. "A Graph Database Representation of Portuguese Criminal-Related Documents". In: *Informatics* 8.2 (June 2021), p. 37. DOI: 10.3390/informatics8020037. URL: https://doi.org/10.3390/informatics8020037 (cit. on pp. 19, 22, 23, 60).

[9] B. Chandrasekaran, J. Josephson, and V. Benjamins. "What are ontologies, and why do we need them?" In: *IEEE Intelligent Systems* 14.1 (Jan. 1999), pp. 20–26. DOI: 10.1109/5254.747902. URL: https://doi.org/10.1109/5254.747902 (cit. on pp. 18, 34).

[10] M. Cheatham et al. "The GeoLink knowledge graph". In: *Big Earth Data* 2.2 (Apr. 2018), pp. 131–143. DOI: 10.1080/20964471.2018.1469291. URL: https://doi.org/10.1080/20964471.2018.1469291 (cit. on p. 23).

[11] M. Costa, D. Gomes, and M. J. Silva. "The evolution of web archiving". In: *International Journal on Digital Libraries* 18.3 (May 2016), pp. 191–205. DOI: 10.1007/s00799-016-0171-9. URL: https://doi.org/10.1007/s00799-016-0171-9 (cit. on p. 22).

[12] M. Costa and M. J. Silva. "Characterizing Search Behavior in Web Archives". In: *Proceedings of the 1st International Temporal Web Analytics Workshop (TWAW 2011)*. Ed. by R. Baeza-Yates, J. Masanès, and M. Spaniol. Vol. 707. CEUR Workshop Proceedings. Hyderabad, India, Mar. 2011. URL: http://CEUR-WS.org/Vol-707/ (cit. on p. 22).

[13] M. Costa and M. J. Silva. "Evaluating Web Archive Search Systems". In: *Proceedings of the 13th International Conference on Web Information Systems Engineering*. Springer-Verlag, 2012, pp. 440–454. ISBN: 9783642350627. DOI: 10.1007/978-3-642-35063-4_32. URL: https://doi.org/10.1007/978-3-642-35063-4_32 (cit. on p. 22).

[14] M. Damova. "Linked Open Data Prototype of the Historical Archive of the European Commission". In: *Archiving Conference* 17.1 (Apr. 2020), pp. 92–97. DOI: 10.2352/issn.2168-3204.2020.1.0.92. URL: https://doi.org/10.2352/issn.2168-3204.2020.1.0.92 (cit. on p. 24).

[15] S. S. Dasgupta, S. N. Ray, and P. Talukdar. "HyTE: Hyperplane-based Temporally aware Knowledge Graph Embedding". In: *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/d18-1225. URL: https://doi.org/10.18653/v1/d18-1225 (cit. on p. 58).

[16] L. Ehrlinger and W. Wöß. "Towards a Definition of Knowledge Graphs". In: *International Conference on Semantic Systems*. 2016 (cit. on p. 11).

[17] O. Etzioni et al. "Open information extraction from the web". In: *Communications of the ACM* 51.12 (Dec. 2008), pp. 68–74. DOI: 10.1145/1409360.1409378. URL: https://doi.org/10.1145/1409360.1409378 (cit. on p. 8).

[18] P. Fafalios et al. "Building and Querying Semantic Layers for Web Archives (Extended Version)". In: *Int. J. Digit. Libr.* 21.2 (June 2020), pp. 149–167. ISSN: 1432-5012. DOI: 10.1007/s00799-018-0251-0. URL: https://doi.org/10.1007/s00799-018-0251-0 (cit. on pp. 21–23, 30).

[19] P. Fafalios et al. "Tracking the history and evolution of entities: entity-centric temporal analysis of large social media archives". In: *International Journal on Digital Libraries* 21.1 (Oct. 2018), pp. 5–17. DOI: 10.1007/s00799-018-0257-7. URL: https://doi.org/10.1007/s00799-018-0257-7 (cit. on pp. 20, 21, 23, 32).

[20] R. Goel et al. "Diachronic Embedding for Temporal Knowledge Graph Completion". In: *Proceedings of the AAAI Conference on Artificial Intelligence* 34.04 (Apr. 2020), pp. 3988–3995. DOI: 10.1609/aaai.v34i04.5815. URL: https://doi.org/10.1609/aaai.v34i04.5815 (cit. on p. 16).

[21] G. Gossen, T. Risse, and E. Demidova. "Towards extracting event-centric collections from Web archives". In: *International Journal on Digital Libraries* 21.1 (2018), pp. 31–45. DOI: 10.1007/s00799-018-0258-6. URL: https://doi.org/10.1007/s00799-018-0258-6 (cit. on p. 23).

[22] S. Gottschalk and E. Demidova. "EventKG: A Multilingual Event-Centric Temporal Knowledge Graph". In: *The Semantic Web*. Springer International Publishing, 2018, pp. 272–287. DOI: 10.1007/978-3-319-93417-4_18. URL: https://doi.org/10.1007/978-3-319-93417-4_18 (cit. on pp. 23, 31, 60).

[23] R. Grishman and B. Sundheim. "Message Understanding Conference- 6: A Brief History". In: *COLING 1996 Volume 1: The 16th International Conference on Computational Linguistics*. 1996. URL: https://aclanthology.org/C96-1079 (cit. on p. 7).

[24] W. R. van Hage et al. "Design and use of the Simple Event Model (SEM)". In: *Journal of Web Semantics* 9.2 (July 2011), pp. 128–136. DOI: 10.1016/j.websem.2011.03.003. URL: https://doi.org/10.1016/j.websem.2011.03.003 (cit. on pp. 19, 31).

[25] A. Hawkins. "Archives, linked data and the digital humanities: increasing access to digitised and born-digital archives via the semantic web". In: *Archival Science* 22.3 (Dec. 2021), pp. 319–344. DOI: 10.1007/s10502-021-09381-0. URL: https://doi.org/10.1007/s10502-021-09381-0 (cit. on p. 24).

[26] A. Hogan et al. "Knowledge Graphs". In: *ACM Computing Surveys* 54.4 (May 2022), pp. 1–37. DOI: 10.1145/3447772. URL: https://doi.org/10.1145/3447772 (cit. on pp. 11, 16, 21).

[27] F. Hogenboom et al. "A Survey of event extraction methods from text for decision support systems". In: *Decision Support Systems* 85 (May 2016), pp. 12–22. DOI: 10.1016/j.dss.2016.02.006. URL: https://doi.org/10.1016/j.dss.2016.02.006 (cit. on p. 8).

[28] H. Holzmann and T. Risse. "Accessing web archives from different perspectives with potential synergies". In: *Researchers, practitioners and their use of the archived web*. School of Advanced Study, University of London, 2017. DOI: 10.14296/resaw.0001. URL: https://doi.org/10.14296/resaw.0001 (cit. on p. 23).

[29] X. Huang et al. "Knowledge Graph Embedding Based Question Answering". In: *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*. ACM, Jan. 2019. DOI: 10.1145/3289600.3290956. URL: https://doi.org/10.1145/3289600.3290956 (cit. on p. 20).

[30] L. Jean-Louis, R. Besançon, and O. Ferret. "Using Temporal Cues for Segmenting Texts into Events". In: *Advances in Natural Language Processing*. Springer Berlin Heidelberg, 2010, pp. 150–161. DOI: 10.1007/978-3-642-14770-8_18. URL: https://doi.org/10.1007/978-3-642-14770-8_18 (cit. on p. 8).

[31] D. Khurana et al. "Natural language processing: state of the art, current trends and challenges". In: *Multimedia Tools and Applications* (July 2022). DOI: 10.1007/s11042-022-13428-4. URL: https://doi.org/10.1007/s11042-022-13428-4 (cit. on p. 6).

[32] M. Kusner et al. "From Word Embeddings To Document Distances". In: *Proceedings of the 32nd International Conference on Machine Learning*. Vol. 37. Proceedings of Machine Learning Research. Lille, France: PMLR, July 2015, pp. 957–966. URL: https://proceedings.mlr.press/v37/kusnerb15.html (cit. on p. 9).

[33] J. Lehmann et al. "DBpedia – A large-scale, multilingual knowledge base extracted from Wikipedia". In: *Semantic Web* 6.2 (2015), pp. 167–195. DOI: 10.3233/sw-140134. URL: https://doi.org/10.3233/sw-140134 (cit. on pp. 19, 23).

[34] S. Lieber et al. "BESOCIAL: A Sustainable Knowledge Graph-Based Workflow for Social Media Archiving". In: *Studies on the Semantic Web*. IOS Press, Aug. 2021. DOI: 10.3233/ssw210045. URL: https://doi.org/10.3233/ssw210045 (cit. on pp. 23, 30).

[35] J. Lin, M. Gholami, and J. Rao. "Infrastructure for Supporting Exploration and Discovery in Web Archives". In: WWW '14 Companion. New York, NY, USA: Association for Computing Machinery, 2014, pp. 851–856. ISBN: 9781450327459. DOI: 10.1145/2567948.2579045. URL: https://doi.org/10.1145/2567948.2579045 (cit. on p. 23).

[36] R. Lopes. "Rich Large-Scale Portuguese Language Models from Large Portuguese Corpora". MSc diss. NOVA School of Science and Technology, Oct. 2023 (cit. on p. 74).

[37] G. Marchionini. "Exploratory Search: From Finding to Understanding". In: *Commun. ACM* 49.4 (2006), pp. 41–46. ISSN: 0001-0782. DOI: 10.1145/1121949.1121979. URL: https://doi.org/10.1145/1121949.1121979 (cit. on p. 22).

[38] P. Marques Alves and C. Levezinho. *A estratégia de comunicação do PCP na internet: promoção de novos espaços de interação?* 2021 (cit. on p. 2).

[39] L. Màrquez et al. "Semantic Role Labeling: An Introduction to the Special Issue". In: *Computational Linguistics* 34.2 (June 2008), pp. 145–159. DOI: 10.1162/coli.2008.34.2.145. URL: https://doi.org/10.1162/coli.2008.34.2.145 (cit. on p. 8).

[40] P. J. P. Martins, L. J. A. D. Costa, and J. C. Ramalho. "Knowledge Graph of press clippings referring social minorities". In: *CEUR Workshop Proceedings*. 2021 (cit. on p. 48).

[41] W. Medhat, A. Hassan, and H. Korashy. "Sentiment analysis algorithms and applications: A survey". In: *Ain Shams Engineering Journal* 5.4 (Dec. 2014), pp. 1093–1113. DOI: 10.1016/j.asej.2014.04.011. URL: https://doi.org/10.1016/j.asej.2014.04.011 (cit. on p. 9).

[42] T. Mikolov et al. "Efficient Estimation of Word Representations in Vector Space". In: *Proceedings of Workshop at ICLR*. 2013 (cit. on p. 9).

[43] G. A. Miller. "WordNet: A Lexical Database for English". In: *Commun. ACM* 38.11 (1995), pp. 39–41. ISSN: 0001-0782. DOI: 10.1145/219717.219748. URL: https://doi.org/10.1145/219717.219748 (cit. on p. 24).

[44] "The Oxford Handbook of Computational Linguistics 2nd edition". In: (Apr. 2014). Ed. by R. Mitkov. DOI: 10.1093/oxfordhb/9780199573691.001.0001. URL: https://doi.org/10.1093/oxfordhb/9780199573691.001.0001 (cit. on p. 7).

[45] T. Al-Moslmi et al. "Named Entity Extraction for Knowledge Graphs: A Literature Overview". In: *IEEE Access* 8 (2020), pp. 32862–32881. DOI: 10.1109/access.2020.2973928. URL: https://doi.org/10.1109/access.2020.2973928 (cit. on p. 7).

[46] J. Niu. "Functionalities of Web Archives". In: *D Lib Mag.* 18 (2012) (cit. on p. 22).

[47] A. Ntoulas, J. Cho, and C. Olston. "What's new on the web?" In: *Proceedings of the 13th international conference on World Wide Web*. ACM, 2004. DOI: 10.1145/988672.988674. URL: https://doi.org/10.1145/988672.988674 (cit. on p. 22).

[48] M. Ostendorff et al. "Enriching BERT with Knowledge Graph Embeddings for Document Classification". In: *ArXiv* abs/1909.08402 (2019). URL: https://api.semanticscholar.org/CorpusID:202661068 (cit. on pp. 20, 59).

[49] K. R. Page et al. "Realising a Layered Digital Library: Exploration and Analysis of the Live Music Archive through Linked Data". In: *2017 ACM/IEEE Joint Conference on Digital Libraries (JCDL)*. IEEE, 2017. DOI: 10.1109/jcdl.2017.7991563. URL: https://doi.org/10.1109/jcdl.2017.7991563 (cit. on p. 24).

[50] H. Paulheim. "Knowledge graph refinement: A survey of approaches and evaluation methods". In: *Semantic Web* 8.3 (Dec. 2016). Ed. by P. Cimiano, pp. 489–508. DOI: 10.3233/sw-160218. URL: https://doi.org/10.3233/sw-160218 (cit. on p. 11).

[51] V. Peña-Araya et al. "HyperStorylines: Interactively untangling dynamic hypergraphs". In: *Information Visualization* 21.1 (2022), pp. 38–62. DOI: 10.1177/14738716211045007. URL: https://doi.org/10.1177/14738716211045007 (cit. on pp. 3, 74).

[52] J. Pennington, R. Socher, and C. Manning. "Glove: Global Vectors for Word Representation". In: *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*. Association for Computational Linguistics, 2014. DOI: 10.3115/v1/d14-1162. URL: https://doi.org/10.3115/v1/d14-1162 (cit. on p. 9).

[53] M. Peters et al. "Deep Contextualized Word Representations". In: *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*. Association for Computational Linguistics, 2018. DOI: 10.18653/v1/n18-1202. URL: https://doi.org/10.18653/v1/n18-1202 (cit. on p. 10).

[54] P. Quaresma et al. "Event Extraction and Representation: A Case Study for the Portuguese Language". In: *Information* 10.6 (June 2019), p. 205. DOI: 10.3390/info10060205. URL: https://doi.org/10.3390/info10060205 (cit. on p. 8).

[55] A. Rademaker et al. "A linked open data architecture for the historical archives of the Getulio Vargas Foundation". In: *International Journal on Digital Libraries* 15.2-4 (Mar. 2015), pp. 153–167. DOI: 10.1007/s00799-015-0147-1. URL: https://doi.org/10.1007/s00799-015-0147-1 (cit. on p. 24).

[56] K. Raiyani et al. "Automated Event Extraction Model for Linked Portuguese Documents". In: *Text2Story@ECIR*. 2019 (cit. on pp. 31, 32).

[57] *RDF 1.1 Concepts and Abstract Syntax — w3.org*. https://www.w3.org/TR/rdf11-concepts/. [Accessed 10-Jan-2023] (cit. on p. 12).

[58] N. Reimers and I. Gurevych. "Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks". In: *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. Association for Computational Linguistics, 2019. DOI: 10.18653/v1/d19-1410. URL: https://doi.org/10.18653/v1/d19-1410 (cit. on p. 10).

[59] M. Rospocher et al. "Building event-centric knowledge graphs from news". In: *Journal of Web Semantics* 37-38 (Mar. 2016), pp. 132–151. DOI: 10.1016/j.websem.2015.12.004. URL: https://doi.org/10.1016/j.websem.2015.12.004 (cit. on pp. 16, 20, 21, 23).

[60] B. Ruddock. "Linked Data and the LOCAH project". In: *Business Information Review* 28.2 (June 2011), pp. 105–111. DOI: 10.1177/0266382111404013. URL: https://doi.org/10.1177/0266382111404013 (cit. on p. 24).

[61] S. Russell and P. Norvig. *Artificial intelligence*. 4th ed. Philadelphia, PA: Pearson Education, Apr. 2020 (cit. on p. 7).

[62] E. W. Schneider. "Course Modularization Applied: The Interface System and Its Implications For Sequence Control and Data Analysis." In: 1973 (cit. on p. 10).

[63] W. Shen, J. Wang, and J. Han. "Entity Linking with a Knowledge Base: Issues, Techniques, and Solutions". In: *IEEE Transactions on Knowledge and Data Engineering* 27.2 (Feb. 2015), pp. 443–460. DOI: 10.1109/tkde.2014.2327028. URL: https://doi.org/10.1109/tkde.2014.2327028 (cit. on p. 7).

[64] A. Silberschatz, H. Korth, and S. Sudarshan. "Database System Concepts". In: 7th ed. New York, NY: McGraw-Hill, Apr. 2019, pp. 8–10 (cit. on pp. 12, 18).

[65] A. Singhal. *Introducing the Knowledge Graph: things, not strings. Google Blog*. https://www.blog.google/products/search/introducing-knowledge-graph-things-not/. [Accessed 12-Dec-2022]. 2012 (cit. on p. 10).

[66] R. Studer, V. Benjamins, and D. Fensel. "Knowledge engineering: Principles and methods". In: *Data & Knowledge Engineering* 25.1-2 (Mar. 1998), pp. 161–197. DOI: 10.1016/s0169-023x(97)00056-6. URL: https://doi.org/10.1016/s0169-023x(97)00056-6 (cit. on p. 18).

[67] F. M. Suchanek, G. Kasneci, and G. Weikum. "Yago: a core of semantic knowledge". In: *The Web Conference*. 2007 (cit. on p. 23).

[68] Y. Sun et al. "PathSim: Meta Path-Based Top-K Similarity Search in Heterogeneous Information Networks". In: *Proc. VLDB Endow.* 4.11 (June 2020), pp. 992–1003. ISSN: 2150-8097. DOI: 10.14778/3402707.3402736. URL: https://doi.org/10.14778/3402707.3402736 (cit. on p. 14).

[69] Z. Sun et al. "RotatE: Knowledge Graph Embedding by Relational Rotation in Complex Space". In: *International Conference on Learning Representations*. 2019. URL: https://openreview.net/forum?id=HkgEQnRqYQ (cit. on p. 57).

[70] UNESCO. *Charter on the Preservation of Digital Heritage — en.unesco.org*. https://en.unesco.org/about-us/legal-affairs/charter-preservation-digital-heritage. [Accessed 04-Jan-2023] (cit. on p. 1).

[71] *Universal Declaration on Archives - UDA | International Council on Archives — ica.org*. https://www.ica.org/en/universal-declaration-archives. [Accessed 04-Jan-2023] (cit. on p. 1).

[72] A. Vaswani et al. "Attention is All You Need". In: *Proceedings of the 31st International Conference on Neural Information Processing Systems*. NIPS'17. Red Hook, NY, USA: Curran Associates Inc., 2017, pp. 6000–6010. ISBN: 9781510860964 (cit. on p. 10).

[73] D. Vrandečić and M. Krötzsch. "Wikidata: A Free Collaborative Knowledgebase". In: *Commun. ACM* 57.10 (Sept. 2014), pp. 78–85. ISSN: 0001-0782. DOI: 10.1145/2629489. URL: https://doi.org/10.1145/2629489 (cit. on pp. 23, 55).

[74] Q. Wang et al. "Knowledge Graph Embedding: A Survey of Approaches and Applications". In: *IEEE Transactions on Knowledge and Data Engineering* 29.12 (Dec. 2017), pp. 2724–2743. DOI: 10.1109/tkde.2017.2754499. URL: https://doi.org/10.1109/tkde.2017.2754499 (cit. on pp. 60, 63).

[75] X. Wang et al. "KEPLER: A Unified Model for Knowledge Embedding and Pre-trained Language Representation". In: *Transactions of the Association for Computational Linguistics* 9 (Mar. 2021), pp. 176–194. DOI: 10.1162/tacl_a_00360. URL: https://doi.org/10.1162/tacl_a_00360 (cit. on p. 57).

[76] J. Webber, I. Robinson, and E. Elfrem. *Graph Databases 2e*. en. Sebastopol, CA: O'Reilly Media, July 2015 (cit. on pp. 15, 74).

[77] G. Weikum et al. "Longitudinal Analytics on Web Archive Data: It's About Time!" In: *5th biennial Conference on Innovative Data Systems Research (CIDR)*. Asilomar, United States, 2011. URL: https://hal.archives-ouvertes.fr/hal-01122664 (cit. on p. 22).

[78] M. Whitelaw. "Generous Interfaces for Digital Cultural Collections". In: *Digit. Humanit. Q.* 9 (2015) (cit. on p. 22).

[79] C. Xu et al. "Temporal Knowledge Graph Completion using a Linear Temporal Regularizer and Multivector Embeddings". In: *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. Online: Association for Computational Linguistics, June 2021, pp. 2569–2578. DOI: 10.18653/v1/2021.naacl-main.202. URL: https://aclanthology.org/2021.naacl-main.202 (cit. on pp. 16, 20, 22).

[80] V. Yadav and S. Bethard. "A Survey on Recent Advances in Named Entity Recognition from Deep Learning models". In: *Proceedings of the 27th International Conference on Computational Linguistics*. Santa Fe, New Mexico, USA: Association for Computational Linguistics, Aug. 2018, pp. 2145–2158. URL: https://aclanthology.org/C18-1182 (cit. on p. 7).

[81] L. Yang et al. "Dynamic Heterogeneous Graph Embedding Using Hierarchical Attentions". In: *Lecture Notes in Computer Science*. Springer International Publishing, 2020, pp. 425–432. DOI: 10.1007/978-3-030-45442-5_53. URL: https://doi.org/10.1007/978-3-030-45442-5_53 (cit. on p. 14).

[82] C.-m. A. Yeung and A. Jatowt. "Studying how the past is remembered". In: *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, Oct. 2011. DOI: 10.1145/2063576.2063755. URL: https://doi.org/10.1145/2063576.2063755 (cit. on p. 51).

[83] G. Zhou et al. "Exploring Various Knowledge in Relation Extraction". In: *Proceedings of the 43rd Annual Meeting of the Association for Computational Linguistics (ACL'05)*. Ann Arbor, Michigan: Association for Computational Linguistics, June 2005, pp. 427–434. DOI: 10.3115/1219840.1219893. URL: https://aclanthology.org/P05-1053 (cit. on p. 7).

I

Full list of filtered domains

publico.pt
jornal.publico.pt
inimigo.publico.pt
m.publico.pt
ultimahora.publico.pt
cinecartaz.publico.pt
m.cinecartaz.publico.pt
ecosfera.publico.clix.pt
assinaturas.publico.pt
blogs.publico.pt
pet.publico.pt
coleccoes.publico.pt
sequeira.publico.pt
i.publico.pt
dn.pt
expresso.pt
lazer.publico.pt
static.publico.pt
economia.publico.pt
movel.publico.pt
emprego.publico.pt
lifestyle.publico.pt
ipsilon.publico.pt
p3.publico.pt
mobile.publico.pt
m.lazer.publico.pt
cidades.publico.pt
acervo.publico.pt
uol.publico.pt
erro.publico.pt
observador.pt

II

Full result tables

Table II.1: Evaluation results of WSA graphs, using the Flat Average weighting method

			WSA				
Weighting	**Method**	**Metric**	**All**	**Pol**	**Spo**	**H&S**	**B&E**
Flat Average	Best@1	Accuracy $\Delta = 0$	0.278	0.321	0.182	0.407	0.25
		Accuracy $\Delta = 1$	0.617	0.643	0.545	0.704	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0	0	0	0	0
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0
		$F_1(5)$	0.435	0.486	0.308	0.579	0.4
	Best@10	Accuracy $\Delta = 0$	0.278	0.321	0.227	0.407	0.375
		Accuracy $\Delta = 1$	0.617	0.643	0.591	0.704	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0	0	0	0	0
		$F_1(3)$	0	0	0.286	0	0
		$F_1(4)$	0	0	0	0	0.462
		$F_1(5)$	0.435	0.486	0.308	0.579	0.4
	Average	Accuracy $\Delta = 0$	0.278	0.321	0.182	0.407	0.333
		Accuracy $\Delta = 1$	0.617	0.643	0.5	0.704	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0	0	0	0	0
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0.364
		$F_1(5)$	0.435	0.486	0.308	0.579	0.429
	Across Years	Accuracy $\Delta = 0$	0.278	0.321	0.227	0.407	0.375
		Accuracy $\Delta = 1$	0.617	0.643	0.591	0.704	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0	0	0	0	0
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0.5
		$F_1(5)$	0.435	0.486	0.4	0.579	0.444
	Expanding Window	Accuracy $\Delta = 0$	0.075	0.105	0.0	0.042	0.115
		Accuracy $\Delta = 1$	0.075	0.105	0.0	0.042	0.115
	Scoped Average	Accuracy $\Delta = 0$	0.113	0.105	0.111	0.083	0.077
		Accuracy $\Delta = 1$	0.113	0.105	0.111	0.083	0.115

Table II.2: Evaluation results of WSA graphs, using the Weighted Average weighting method

Weighting	Method	Metric	WSA Graphs All	Pol	Spo	H&S	B&E
Weighted Average	Best@1	Accuracy $\Delta = 0$	0.278	0.286	0.273	0.37	0.208
		Accuracy $\Delta = 1$	0.643	0.607	0.636	0.667	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0	0	0	0	0
		$F_1(3)$	0.296	0.308	0.4	0	0.154
		$F_1(4)$	0.05	0	0	0	0
		$F_1(5)$	0.418	0.387	0.353	0.606	0.381
	Best@10	Accuracy $\Delta = 0$	0.365	0.464	0.409	0.407	0.333
		Accuracy $\Delta = 1$	0.765	0.786	0.727	0.815	0.792
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0.167	0	0	0	0
		$F_1(3)$	0.269	0.438	0.37	0.095	0.308
		$F_1(4)$	0.095	0	0.25	0	0
		$F_1(5)$	0.418	0.387	0.353	0.606	0.381
	Average	Accuracy $\Delta = 0$	0.278	0.321	0.182	0.407	0.333
		Accuracy $\Delta = 1$	0.617	0.643	0.5	0.704	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0	0	0	0	0
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0.364
		$F_1(5)$	0.435	0.486	0.32	0.579	0.429
	Across Years	Accuracy $\Delta = 0$	0.287	0.429	0.227	0.296	0.208
		Accuracy $\Delta = 1$	0.739	0.821	0.591	0.741	0.625
		$F_1(1)$	0	0	0	0	0
		$F_1(2)$	0.154	0	0	0	0
		$F_1(3)$	0.313	0.526	0.364	0	0.333
		$F_1(4)$	0.437	0.56	0.222	0.609	0.267
		$F_1(5)$	0	0	0	0.167	0
	Expanding Window	Accuracy $\Delta = 0$	0.019	0.053	0.056	0.042	0.038
		Accuracy $\Delta = 1$	0.038	0.105	0.056	0.042	0.038
	Scoped Average	Accuracy $\Delta = 0$	0.066	0.105	0.056	0.125	0.038
		Accuracy $\Delta = 1$	0.075	0.105	0.056	0.125	0.038

Table II.3: Evaluation results of WPS graphs, using the Flat Average weighting method

			WPS Graphs				
Weighting	**Method**	**Metric**	**All**	**Pol**	**Spo**	**H&S**	**B&E**
Flat Average	Best@1	Accuracy $\Delta = 0$	0.283	0.333	0.269	0.357	0.192
		Accuracy $\Delta = 1$	0.717	0.7	0.654	0.714	0.769
		$F_1(1)$	0.125	0.4	0	0	0
		$F_1(2)$	0.4	0	0	0	0
		$F_1(3)$	0.3	0.211	0.235	0	0.267
		$F_1(4)$	0.179	0.375	0.353	0.3	0
		$F_1(5)$	0.366	0.444	0.364	0.56	0.375
	Best@10	Accuracy $\Delta = 0$	0.492	0.5	0.423	0.571	0.423
		Accuracy $\Delta = 1$	0.792	0.833	0.731	0.821	0.808
		$F_1(1)$	0.095	0.333	0	0	0
		$F_1(2)$	0.24	0	0.25	0.222	0
		$F_1(3)$	0.323	0.357	0.308	0.118	0.32
		$F_1(4)$	0.453	0.417	0.364	0.424	0.32
		$F_1(5)$	0.366	0.444	0.364	0.56	0.375
	Average	Accuracy $\Delta = 0$	0.275	0.267	0.308	0.286	0.269
		Accuracy $\Delta = 1$	0.717	0.767	0.654	0.786	0.692
		$F_1(1)$	0.118	0.4	0	0	0
		$F_1(2)$	0.2	0	0.333	0	0
		$F_1(3)$	0.338	0.32	0.381	0.182	0.25
		$F_1(4)$	0.293	0.222	0.167	0.421	0.353
		$F_1(5)$	0.235	0.2	0.5	0.375	0.444
	Across Years	Accuracy $\Delta = 0$	0.15	0.133	0.115	0.143	0.154
		Accuracy $\Delta = 1$	0.35	0.4	0.346	0.429	0.423
		$F_1(1)$	0.429	0.6	0.182	0.6	0.6
		$F_1(2)$	0.116	0.111	0.211	0.118	0.118
		$F_1(3)$	0.1	0	0	0	0
		$F_1(4)$	0	0	0	0	0
		$F_1(5)$	0	0	0	0	0
	Expanding Window	Accuracy $\Delta = 0$	0.047	0.079	0.0	0.042	0.0
		Accuracy $\Delta = 1$	0.085	0.132	0.111	0.083	0.0
	Scoped Average	Accuracy $\Delta = 0$	0.019	0.026	0.111	0.0	0.0
		Accuracy $\Delta = 1$	0.019	0.026	0.111	0.0	0.0

Table II.4: Evaluation results of WPS graphs, using the Weighted Average weighting method

Weighting	Method	Metric	WPS Graphs All	Pol	Spo	H&S	B&E
Weighted Average	Best@1	Accuracy $\Delta = 0$	0.2	0.167	0.269	0.143	0.231
		Accuracy $\Delta = 1$	0.617	0.533	0.654	0.571	0.692
		$F_1(1)$	0.118	0.4	0	0	0
		$F_1(2)$	0.176	0	0.333	0	0
		$F_1(3)$	0.253	0.174	0.417	0	0.286
		$F_1(4)$	0.125	0.167	0	0.308	0.182
		$F_1(5)$	0.222	0.167	0.25	0.25	0.333
	Best@10	Accuracy $\Delta = 0$	0.25	0.267	0.269	0.25	0.269
		Accuracy $\Delta = 1$	0.642	0.633	0.654	0.607	0.692
		$F_1(1)$	0.118	0.182	0	0	0
		$F_1(2)$	0.125	0	0.182	0.25	0
		$F_1(3)$	0.248	0.345	0.385	0	0.222
		$F_1(4)$	0.192	0.167	0	0.4	0.286
		$F_1(5)$	0.222	0.167	0.25	0.25	0.333
	Average	Accuracy $\Delta = 0$	0.217	0.2	0.308	0.143	0.269
		Accuracy $\Delta = 1$	0.658	0.633	0.654	0.679	0.692
		$F_1(1)$	0.111	0.4	0	0	0
		$F_1(2)$	0.105	0	0.286	0.222	0
		$F_1(3)$	0.225	0.364	0.5	0	0.25
		$F_1(4)$	0.281	0.133	0.167	0.222	0.353
		$F_1(5)$	0.245	0	0.286	0.154	0.444
	Across Years	Accuracy $\Delta = 0$	0.142	0.167	0.115	0.143	0.115
		Accuracy $\Delta = 1$	0.242	0.2	0.231	0.25	0.269
		$F_1(1)$	0.259	0.25	0.273	0.32	0.286
		$F_1(2)$	0.133	0.667	0	0	0
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0
		$F_1(5)$	0	0	0	0	0
	Expanding Window	Accuracy $\Delta = 0$	0.047	0.026	0.0	0.0	0.0
		Accuracy $\Delta = 1$	0.075	0.079	0.0	0.0	0.0
	Scoped Average	Accuracy $\Delta = 0$	0.019	0.053	0.0	0.0	0.0
		Accuracy $\Delta = 1$	0.019	0.053	0.0	0.0	0.0

Table II.5: Evaluation results of LPS graphs, using the Flat Average weighting method

			LPS Graphs				
Weighting	**Method**	**Metric**	**All**	**Pol**	**Spo**	**H&S**	**B&E**
Flat Average	Best@1	Accuracy $\Delta = 0$	0.36	0.375	0.353	0.238	0.318
		Accuracy $\Delta = 1$	0.674	0.792	0.765	0.619	0.682
		$F_1(1)$	0.167	0	0	0	0
		$F_1(2)$	0.125	0	0.222	0	0
		$F_1(3)$	0.407	0.5	0.4	0.4	0.364
		$F_1(4)$	0.408	0.308	0.6	0.133	0.462
		$F_1(5)$	0.383	0.444	0	0.4	0.308
	Best@10	Accuracy $\Delta = 0$	0.506	0.708	0.471	0.429	0.727
		Accuracy $\Delta = 1$	0.775	0.875	0.882	0.81	0.864
		$F_1(1)$	0.105	0.8	0	0	0.333
		$F_1(2)$	0.2	0	0.308	0.25	0.267
		$F_1(3)$	0.338	0.381	0.375	0.118	0.286
		$F_1(4)$	0.514	0.636	0.545	0.364	0.762
		$F_1(5)$	0.383	0.444	0	0.4	0.308
	Average	Accuracy $\Delta = 0$	0.247	0.375	0.235	0.143	0.227
		Accuracy $\Delta = 1$	0.629	0.75	0.824	0.667	0.636
		$F_1(1)$	0.125	0.5	0	0	0
		$F_1(2)$	0.095	0	0.444	0	0
		$F_1(3)$	0.305	0.333	0.333	0.143	0.182
		$F_1(4)$	0.24	0.462	0	0.333	0.5
		$F_1(5)$	0.312	0.364	0	0	0
	Across Years	Accuracy $\Delta = 0$	0.135	0.083	0.176	0.19	0.273
		Accuracy $\Delta = 1$	0.326	0.292	0.471	0.429	0.364
		$F_1(1)$	0.361	0.5	0.2	0.667	0.75
		$F_1(2)$	0.043	0	0.364	0.118	0.3
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0
		$F_1(5)$	0	0	0	0	0
	Expanding Window	Accuracy $\Delta = 0$	0.028	0.026	0.0	0.042	0.038
		Accuracy $\Delta = 1$	0.066	0.079	0.0	0.042	0.077
	Scoped Average	Accuracy $\Delta = 0$	0.009	0.026	0.0	0.0	0.0
		Accuracy $\Delta = 1$	0.019	0.026	0.0	0.0	0.0

Table II.6: Evaluation results of LPS graphs, using the Weighted Average weighting method

Weighting	Method	Metric	LPS Graphs All	Pol	Spo	H&S	B&E
Weighted Average	Best@1	Accuracy $\Delta = 0$	0.292	0.167	0.176	0.143	0.273
		Accuracy $\Delta = 1$	0.685	0.667	0.824	0.524	0.682
		$F_1(1)$	0.154	0	0	0	0
		$F_1(2)$	0.167	0	0.25	0.286	0
		$F_1(3)$	0.29	0.286	0.286	0	0.267
		$F_1(4)$	0.308	0.182	0	0	0.5
		$F_1(5)$	0.424	0.133	0	0.333	0.25
	Best@10	Accuracy $\Delta = 0$	0.393	0.5	0.294	0.238	0.591
		Accuracy $\Delta = 1$	0.73	0.75	0.882	0.571	0.864
		$F_1(1)$	0.083	0.571	0	0.286	0.364
		$F_1(2)$	0.192	0	0.308	0.25	0.222
		$F_1(3)$	0.273	0.348	0.375	0	0.348
		$F_1(4)$	0.435	0.625	0	0	0.5
		$F_1(5)$	0.424	0.133	0	0.333	0.25
	Average	Accuracy $\Delta = 0$	0.191	0.292	0.294	0.095	0.182
		Accuracy $\Delta = 1$	0.629	0.708	0.824	0.571	0.727
		$F_1(1)$	0.125	0.5	0	0	0.5
		$F_1(2)$	0.129	0	0.444	0	0
		$F_1(3)$	0.207	0.286	0.462	0	0.143
		$F_1(4)$	0.14	0.429	0	0.364	0.333
		$F_1(5)$	0.333	0.222	0	0	0
	Across Years	Accuracy $\Delta = 0$	0.135	0.125	0.176	0.143	0.227
		Accuracy $\Delta = 1$	0.258	0.208	0.412	0.381	0.318
		$F_1(1)$	0.244	0.273	0.154	0.353	0.353
		$F_1(2)$	0.095	0	0.5	0	0.364
		$F_1(3)$	0	0	0	0	0
		$F_1(4)$	0	0	0	0	0
		$F_1(5)$	0	0	0	0	0
	Expanding Window	Accuracy $\Delta = 0$	0.028	0.026	0.0	0.0	0.038
		Accuracy $\Delta = 1$	0.057	0.105	0.0	0.042	0.115
	Scoped Average	Accuracy $\Delta = 0$	0.019	0.026	0.0	0.0	0.0
		Accuracy $\Delta = 1$	0.028	0.026	0.0	0.0	0.0

Table II.7: Evaluation results of the WSA^{wiki} graphs, using the Flat Average weighting method

Weighting	Method	Metric	WSA^{wiki} Graph All
Flat Average	Best@1	Accuracy $\Delta = 0$	0.278
		Accuracy $\Delta = 1$	0.617
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0
		$F_1(4)$	0
		$F_1(5)$	0.435
	Best@10	Accuracy $\Delta = 0$	0.278
		Accuracy $\Delta = 1$	0.617
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0
		$F_1(4)$	0
		$F_1(5)$	0.435
	Average	Accuracy $\Delta = 0$	0.278
		Accuracy $\Delta = 1$	0.617
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0
		$F_1(4)$	0
		$F_1(5)$	0.435
	Across Years	Accuracy $\Delta = 0$	0.278
		Accuracy $\Delta = 1$	0.617
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0
		$F_1(4)$	0
		$F_1(5)$	0.435
	Expanding Window	Accuracy $\Delta = 0$	0.075
		Accuracy $\Delta = 1$	0.075
	Scoped Average	Accuracy $\Delta = 0$	0.113
		Accuracy $\Delta = 1$	0.113

Table II.8: Evaluation results of the WSA^{wiki} graphs, using the Weighted Average weighting method

Weighting	Method	Metric	WSA^{wiki} Graph All
Weighted Average	Best@1	Accuracy $\Delta = 0$	0.261
		Accuracy $\Delta = 1$	0.643
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0.303
		$F_1(4)$	0
		$F_1(5)$	0.404
	Best@10	Accuracy $\Delta = 0$	0.322
		Accuracy $\Delta = 1$	0.739
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0.271
		$F_1(4)$	0.049
		$F_1(5)$	0.404
	Average	Accuracy $\Delta = 0$	0.278
		Accuracy $\Delta = 1$	0.617
		$F_1(1)$	0
		$F_1(2)$	0
		$F_1(3)$	0
		$F_1(4)$	0
		$F_1(5)$	0.435
	Across Years	Accuracy $\Delta = 0$	0.226
		Accuracy $\Delta = 1$	0.661
		$F_1(1)$	0
		$F_1(2)$	0.154
		$F_1(3)$	0.267
		$F_1(4)$	0.338
		$F_1(5)$	0
	Expanding Window	Accuracy $\Delta = 0$	0.028
		Accuracy $\Delta = 1$	0.066
	Scoped Average	Accuracy $\Delta = 0$	0.075
		Accuracy $\Delta = 1$	0.094