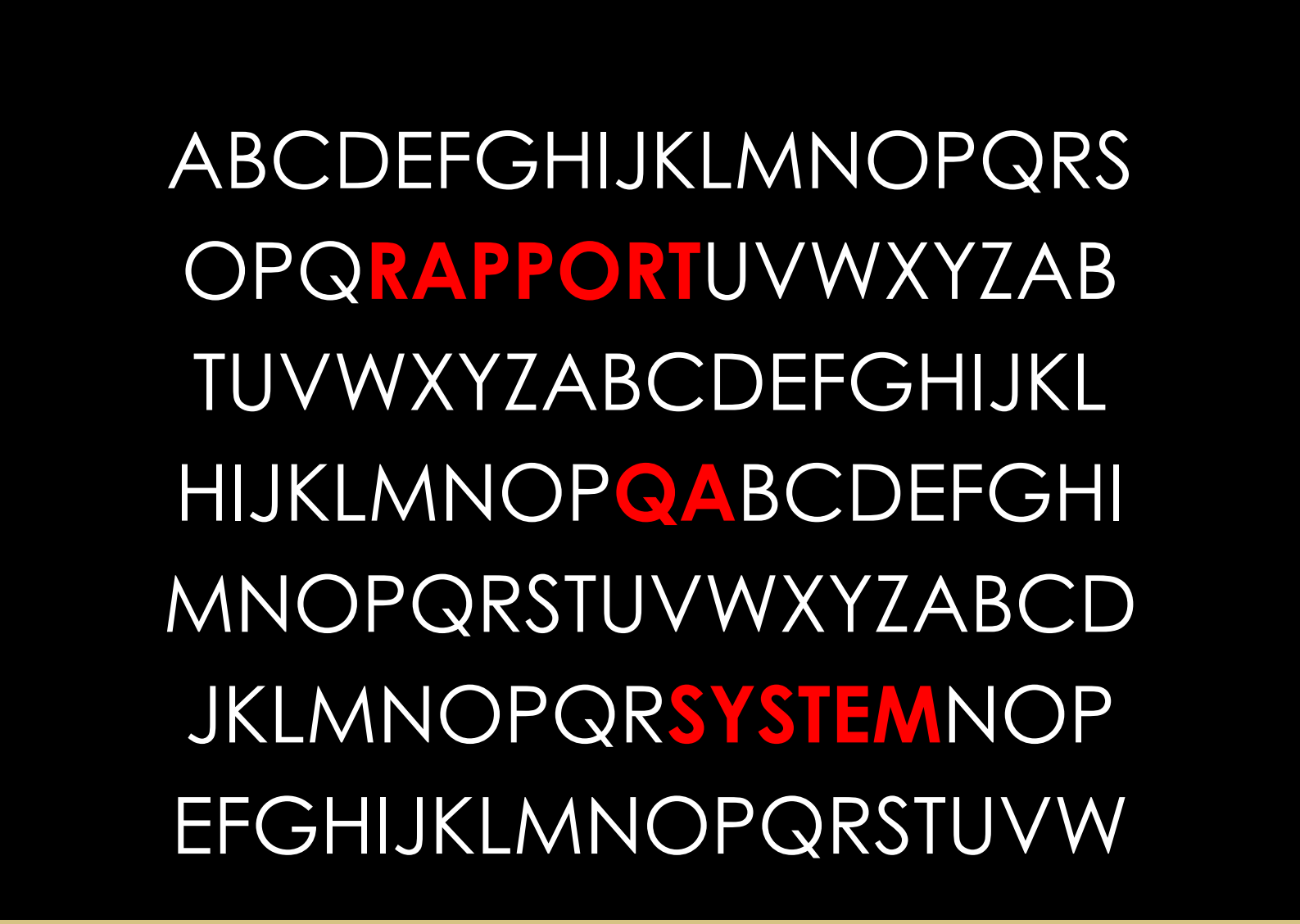

Ricardo Manuel da Conceição Rodrigues

RAPPort: A Fact-Based Question Answering System for Portuguese

Ph.D. Thesis in Information Science and Technology, supervised by Professor Paulo Jorge de Sousa Gomes and Professor Fernando Jorge Penousal Martins Machado, submitted to the Department of Informatics Engineering of the Faculty of Sciences and Technology of the University of Coimbra

June 2017

Universidade de Coimbra

Ph.D. Thesis
Doctoral Program in Information Science and Technology
Artificial Intelligence

RAPPORT: A Fact-Based Question Answering System for Portuguese

Ricardo Manuel da Conceição Rodrigues

Thesis Advisors
Paulo Jorge de Sousa Gomes
Fernando Jorge Penousal Martins Machado

Department of Informatics Engineering
Faculty of Sciences and Technology
University of Coimbra

June 2017

Acknowledgements

I would like to thank both Prof. Dr. Paulo Gomes and Prof. Dr. Fernando Penousal Machado, my advisors at distinct times, for their support and constant suggestions over the course of the work that resulted in this thesis, and of the writing of the thesis itself. Paulo Gomes allowed me a first contact with the topic of this thesis, and always supported me in all the stages of the research work and of this thesis, even after leaving Academia for pursuing a full time job as the chief executive officer of Wizdee. Fernando Penousal Machado stepped in in the final stages of this work, helping me out fine tuning the last issues with the thesis.

I must also express my gratitude to the members of the Knowledge and Intelligent Systems (KIS) laboratory of the Cognitive and Media Systems (CMS) group, an integrant part of the Center for Informatics and Systems of the University of Coimbra (CISUC), with a due and special token of appreciation going to Hugo Gonçalo Oliveira, for all the help and work shared on multiple projects.

A special thanks goes to my parents for all their patience and support, providing the basis for all I have achieved. Without them, all this work, and, most of all, all that preceded it and about everything else, would not have been possible.

I would also want to thank the friends who have accompanied me in *Lusa Atenas*, specially Marco Veloso, Nuno Gil, João Cunha and Miguel Silva, for all the time we spent together and the fellowship demonstrated over the last near twenty years.

Last, but not least, I want to thank my dearest Joana for everything in our shared life, including her support and strength in all the stages of the writing of this thesis, and both Pedro and Vasco, our children, for being there at the end of the day, always cheering me up.

Abstract

Question answering is one of the longest-standing problems in natural language processing. Although natural language interfaces for computer systems can be considered more common these days, the same still does not happen regarding access to specific textual information. Any full text search engine can easily retrieve documents containing user specified or closely related terms, however it is typically unable to answer user questions with small passages or short answers.

The problem with question answering is that text is hard to process, due to its syntactic structure and, to a higher degree, to its semantic contents. At the sentence level, although the syntactic aspects of natural language have well known rules, the size and complexity of a sentence may make it difficult to analyze its structure. Furthermore, semantic aspects are still arduous to address, with text ambiguity being one of the hardest tasks to handle. There is also the need to correctly process the question in order to define its target, and then select and process the answers found in a text. Additionally, the selected text that may yield the answer to a given question must be further processed in order to present just a passage instead of the full text. These issues take also longer to address in languages other than English, as is the case of Portuguese, that have a lot less people working on them.

This work focuses on question answering for Portuguese. In other words, our field of interest is in the presentation of short answers, passages, and possibly full sentences, but not whole documents, to questions formulated using natural language. For that purpose, we have developed a system, RAPPORT, built upon the use of open information extraction techniques for extracting triples, so called *facts*, characterizing information on text files, and then storing and using them for answering user queries done in natural language. These facts, in the form of *subject*, *predicate* and *object*, alongside other metadata, constitute the basis of the answers presented by the system. Facts work both by storing short and direct information found in a text, typically entity related information, and by containing in themselves the answers to the questions already in the form of small passages. As for the results, although there is margin for improvement, they are a tangible proof of the adequacy of our approach and its different modules for storing information and retrieving answers in question answering systems.

In the process, in addition to contributing with a new approach to question answering for Portuguese, and validating the application of open information extraction to question answering, we have developed a set of tools that has been used in other natural language processing related works, such as is the case of a lemmatizer, LEMPORT, which was built from scratch, and has a high accuracy. Many of these tools result from the improvement of those found in the *Apache OpenNLP* toolkit, by pre-processing their input, post-processing their output, or both, and by training models for use in those tools or other, such as *MaltParser*. Other tools include the creation of interfaces for other resources containing, for example, synonyms, hypernyms, hyponyms, or the creation of lists of, for instance, relations between verbs and agents, using rules.

Resumo

A resposta automática a perguntas é um dos problemas existentes de há longa data na área de processamento de linguagem natural. Embora interfaces em linguagem natural para sistemas informáticos possam ser consideradas mais comuns hoje em dia, o mesmo ainda não acontece no que toca ao acesso a informação específica em formato textual. Qualquer motor de busca de texto integral pode facilmente recuperar documentos que contenham termos especificados pelo utilizador ou estreitamente relacionados, mas é, regra geral, incapaz de responder a perguntas de utilizadores com passagens directas ou pequenos excertos.

O grande problema com a resposta automática a perguntas deve-se a o texto ser difícil de processar, tanto pela sua estrutura sintáctica como, num grau mais elevado, pelo seu conteúdo semântico. Ao nível das frases, embora os aspectos sintácticos da linguagem natural tenham regras bem conhecidas, o tamanho e a complexidade de uma frase podem tornar difícil a análise da sua estrutura. Ainda para mais, aspectos semânticos são ainda difíceis de tratar, com a ambiguidade do texto a ser uma das tarefas mais difíceis de abordar. Há também a necessidade de processar correctamente as questões, a fim de definir os seus alvos, e depois seleccionar e processar as respostas encontradas no texto. Para além disso, o texto seleccionado que pode conter a resposta a uma dada questão deve ainda ser processado de forma a apresentar apenas uma passagem, em vez do texto completo. Estes problemas são ainda de resolução mais lenta noutros idiomas que não o Inglês, como é o caso do Português, dado que têm muito menos pessoas a debruçarem-se sobre eles.

Este trabalho tem como foco a resposta automática a perguntas para o português. Por outras palavras, o nosso campo de interesse situa-se na obtenção de respostas curtas, excertos, ou eventualmente frases, mas não necessariamente documentos inteiros, a perguntas formuladas usando linguagem natural. Para esse efeito, desenvolvemos um sistema, o RAPPORT, baseado em técnicas de extração de informação aberta para a obtenção triplos ou *factos* que descrevam informação existente em texto, passando, de seguida, ao armazenamento e consulta desses mesmos factos para responder a perguntas do utilizador feitas sob a forma de linguagem natural. Estes factos, caracterizados por *sujeito*, *predicado* e *objecto*, para além de outros metadados, constituem a base

para as respostas apresentadas pelo sistema. Tanto funcionam armazenando apenas o que pode ser considerado informação relevante, tipicamente relacionada com entidades, num texto, como contendo respostas a perguntas na forma de passagens curtas. Quanto aos resultados, apesar de ainda haver margem para melhoramentos, são uma constatação da adequação da nossa abordagem e dos respectivos módulos para o armazenamento de informação e para a obtenção de respostas em sistemas de resposta automática a perguntas.

Neste processo, para além de uma nova abordagem para a resposta automática a perguntas para o Português, e da validação da aplicação de factos obtidos através de extracção de informação aberta à resposta automática a perguntas, desenvolvemos ferramentas que têm sido utilizadas noutros trabalhos relacionados com o processamento de linguagem natural, tal como é o caso de um analisador morfológico, o LEMPORT, que foi construído a partir do zero, e possui uma precisão muito elevada. Dessas ferramentas, a maioria delas resulta de melhoramentos efectuados sobre aquelas encontradas no *Apache OpenNLP*, através do pré-processamento sua entrada, pós-processamento da sua saída, ou ambos, e do treino de modelos para utilização nessas ferramentas e outras, tais como o *MaltParser*. Outras ferramentas incluem a criação de interfaces para outros recursos que contêm, por exemplo, sinónimos, hiperónimos, hipónimos, ou a criação de listas de, por exemplo, relações entre verbos e agentes, usando regras.

Table of Contents

List of Figures

List of Tables

List of Algorithms

Chapter 1

Introduction

In this chapter, we provide an introduction to the topic of the thesis, ranging from the need for question answering systems in general to the definition and implementation of the approach used and resulting contributions.

1.1 Motivation

Since the very beginning, many human endeavours have been quests for knowledge: to know *what*, *who*, *when*, *where*, *why* and *how*. The answers to these questions are also the cornerstone of news stories and articles, and represent what can be extracted from a text in order to fully understand and make proper use of it. As such, it is not surprising that large amounts of data are found in the form of natural language text, such as is the case of newspapers, journals, magazines, books, or the omnipresent World Wide Web (WWW). Furthermore, these questions and associated answers also serve to represent most of the information that can be extracted from texts using computational approaches (see, for instance, Shrihari and Li (2000)).

If, at first, most of human knowledge did not exist and had to be created, nowadays knowledge abounds, and the biggest difficulty is arguably to locate and interpret sources of information, freeing us from the burden of having to know "everything" we may need on any topic at a given moment. Today, there are multiple and vast knowledge sources online, free and publicly accessible. The WWW itself, as a whole, and online encyclopedias, as more specific and processed knowledge sources, are the most recognizable examples of this type of knowledge sources.

The data found in natural language texts are largely unstructured and need to be processed in order to retrieve and extract information, tasks which are addressed in natural language processing (NLP) (Jurafsky and Martin, 2008). That, in addition to being a problem in itself, together with the vast amounts of available information, gives rise to a potential paradox: there is "a surfeit of information and a paucity of useful

information" (Edmunds and Morris, 2000). That is to say, too much information is as good as virtually no information at all, unless it can be properly processed for a specific use or task.

In most cases, we find ourselves dependent on text search engines and on a choice of keywords that may, or may not, be appropriate. Generally speaking, we must avoid natural language constructs, and use instead *meaningful* words, which we think may be present in the requested documents, frequently leading us to spend more time dealing with the search query than with the returned information. That information, by its turn, must also be processed by us, selecting whose results might be of interest for the problem at hands.

Question answering (QA) (Strzalkowski and Harabagiu, 2006) addresses this problem by using natural language in the process. When querying a system that provides or retrieves information about a given topic, with its contents in natural language, the user should be able to pose questions using natural language, and not have to care about system specific details. As any user can relate, we should not be limited in our access to information by things such as:

- knowing the best keywords to get an answer to a specific question — as one may know little, if anything, about the topic;
- using system specific syntax in order to interact with it — since it may have great implications at the usability level;
- perusing the multiple documents that may contain the eventual answer — as it can be, and usually is, a highly time-consuming task;
- being limited by the questions someone has compiled before — as those questions (and corresponding answers) might be narrow in scope or not comprehensive.

The problem is that the knowledge sources may be unstructured and not found in an easy way to be accessed. When that happens, some kind of system that is able to interpret our questions and browse texts looking for possible answers would improve user productivity, since about 30% of the workday of a knowledge worker is spent searching for information (Feldman and Sherman, 2001). That is the main motivation for creating a question answering system.

Question answering is one of the earliest problems addressed in NLP (see, for instance, Green Jr. et al. (1961), and Simmons (1965, 1970)), it has seen some renewed interest over the last two decades, and there is a number of known approaches to the problem (Hirschman and Gaizauskas, 2001; Strzalkowski and Harabagiu, 2006).

1.2 The Thesis

Right from the beginning, it was apparent to us that QA, as most NLP applications, is difficult and laborious. Although there is no secret formula to address both issues, at least the latter can be mitigated. For that, like most of the existing approaches, we have opted for offline processing of the texts and information extraction, leaving just the matching and answer retrieving for runtime.

We also believed that question answering could benefit from new approaches, specially when addressing the Portuguese language, and that new techniques that have not been applied to that language could be implemented and assessed. One of the things we wanted to try was a way of storing just relevant information, instead of storing every bit of information found in a text, and, by doing so, improving both the answers presented to the user and the time it may take, essentially by transferring the extraction of short answers to the document text processing stage instead of doing that at a later time, when the user interacts with the question answering system.

We ended up selecting *facts* for such purpose. Facts store currently short excerpts of information about entities, together with metadata which allows to identify the sentence related to the fact, and also the text or document where the fact can be found.

Our approach is based on the assumption that the use of facts would we be valuable in two ways: facts store just the essential about a given subject, and would provide in themselves the short passages that are expected from a question answering system — at least, as defined in evaluations such as the ones from CLEF.[1]

1.3 Scope

Our QA system proposes to answer open domain user questions in Portuguese, using Portuguese corpora. Open domain QA is arguably harder than closed domain QA, but it has, however, less restrictions and can be used on a vaster amount of information sources. These two types of domain are further characterized in terms of free text-based and structured knowledge-based question answering (Mollá and Vicedo, 2007).

At the same time, QA for Portuguese has not seen significant advances in the last years, as can be observed by the low number of publications since the late 2000s. Moreover, existing approaches are not easy to reproduce (and in some cases it is likely to be impossible). That led us to develop a new system from the ground up, although borrowing already proven tools whenever possible.

We have selected Portuguese for a number of reasons, of which the most significant ones are being our native language, and also the fact that QA systems for Portuguese

[1] The CLEF Initiative: `http://www.clef-initiative.eu/` [Accessed: February 2017].

are not so common as the systems for English. For the Portuguese language, although it may seem research on question answering has been somewhat stagnant, without any knowledge of new approaches in the last few years, there is already over a half dozen systems addressing this area of NLP. However, besides the lack of known recent improvements, all the existing approaches have one or more of the following limitations:

- They are proprietary, unavailable, or their use is not free.
- There is a lack of details regarding implementation and performance.
- The techniques are dated or too tied to the corpora for which they were planned and developed.
- The results fall behind those attained for other languages, specifically regarding their overall accuracy.

Bearing this in mind, we have aimed at developing a new approach to question answering for Portuguese, that would get on pair with the best known approaches for that language, and overcome the limitations just stated, bringing it up to date. The approach should be as modular as possible, both for allowing future improvements to be independent as much as possible, and for tools developed in this context to be used in other scenarios, be them other corpora, other types of questions, or even other NLP areas and tasks.

We have decided to use libraries or packages that were readily available for the tasks included in question answering, and could be easily integrated in our approach, mainly when they could hardly be surpassed by our own efforts. When that was not possible, we have implement our own solutions for such tasks. For such reason, all of this work was soon decided to be made publicly available, as we believe in sharing and retrieving to the community, helping others in the process (as others have helped us).

Although the proposed system is flexible enough to work with any given corpora in Portuguese, we have targeted both CHAVE (Santos and Rocha, 2005) and the Portuguese Wikipedia,[2] as they have been used in the most well known evaluations of question answering for Portuguese (among other languages) organized by CLEF (Conference and Labs of the Evaluation Forum): the QA@CLEF tracks, or more precisely the CLEF Multilingual Question Answering Tracks (Magnini et al., 2005; Vallin et al., 2006; Magnini et al., 2007; Giampiccolo et al., 2008; Forner et al., 2009). As such, all results presented here are limited to those two corpora. The Portuguese Wikipedia was also the target corpus of PÁGICO (Santos, 2012), where RAPPORTÁGICO, which can be considered the first iteration of our approach, was tested.

[2] WIKIPÉDIA, *a enciclopédia livre*: http://pt.wikipedia.org/ [Accessed: February 2017].

1.4 Approach

One of the most identifying elements of our approach is the use of facts as the basic unit of information regarding any topic. Facts consists of triples identifying a *subject*, a *predicate*, and an *object*, together with metadata for identifying the facts and associating them to sentences and documents. These facts are then used as the base for answering questions. Our approach ends up sharing similarities with open information extraction (OIE), regarding extraction and storage of information in triples (Gamallo, 2014), and is, to the best of our knowledge, a first in QA targeted at Portuguese.

Our system starts by processing the corpus, and storing facts about or that characterize any entities in the text. Later on, it processes user questions, selecting key elements and characteristics, such as the questions' target, in order to select which facts can be of use. After those facts are selected and retrieved, they are finally processed to present an answer to the user.

This process is decomposed in a combination of four major modules for addressing information extraction from the corpus, namely:

1. **Fact extraction**: sentences found in the corpus are processed in order to have their contents represented as triples (subject, predicate, and object). This is the module with more individual tasks, including sentence splitting, tokenization, part-of-speech tagging, lemmatization, chunking, and dependency parsing. It is also where open information extraction is performed for extracting facts from the sentences in the documents.

2. **Fact storage**: triples, data regarding sentences, and document text and metadata are stored in distinct indices, using open-source information retrieval software, which are then used for querying and retrieving data.

3. **Fact search**: questions are processed in order to characterize them and retrieve the keywords, that are then used for querying sentences and associated triples (searching for facts). Keywords can be used directly or further processed, adding to or replacing them with related words (e.g., synonyms).

4. **Answer retrieving**: selection and ordering of the facts that are used to produce candidate answers, retrieving the expected passages from the triples, which typically are to be found in their subjects or objects.

The output of the system can be configured to present different data in multiple variations, such as just the passage, the sentence or the whole document (and related metadata). The approach can also process questions in batch or interactively, locally or trough the web.

1.5 Research Goals

Our main objective was to provide a new approach to question answering for the Portuguese language in an open domain, exploring techniques such as the use of facts for describing information that could be used for answering questions. It was also intended that the approach should overcome common question answering difficulties at various levels like the inability to parse the question, find text passages or extract answers.

The approach should also be fast, interacting with the user in real-time — namely on the processing of questions, or on the retrieval of answers levels —, with most of the text processing being done offline, prior to user interaction.

1.6 Contributions

RAPPORT itself can be identified as the main contribution of this work, and it will be made freely available (as a whole) in the near future. However, each of its modules and even some of the tools used in them can be considered individually as contributions for the area of natural language processing — with some of the developed tools being already accessible to the community.

Alongside RAPPORT, the following tools have been developed from scratch, or by pre-processing the input, or by post-processing the output of some of the tools available in *Apache OpenNLP*,[3] a machine learning based toolkit for the processing of natural language text, and also, in one case, *MaltParser* (Nivre et al., 2007):

- Sentence splitter: based on *OpenNLP*, post-processing its output for addressing issues such as abbreviations with periods in them;
- Tokenizer: based on *OpenNLP*, post-processing its output for grouping tokens that should be considered only one entity, and splitting apart others, such as verbs and pronouns, that can be found together in Portuguese using hyphens;
- Entity finder: based on *OpenNLP*, with a model created on *Floresta Virgem* (an automatically revised part of the *Floresta Sintá(c)tica* corpus (Afonso et al., 2002)), from Linguateca, a resource center for the computational processing of the Portuguese language;[4]
- POS tagger: based on *OpenNLP*, using as its input the output of the tokenizer;
- Lemmatizer: built from scratch, LEMPORT (Rodrigues et al., 2014) has become arguably the best of its kind for the Portuguese language;

[3] *Apache OpenNLP*: `http://opennlp.apache.org/` [Accessed: February 2017].
[4] Linguateca: `http://www.linguateca.pt` [Accessed: February 2017].

- Phrase chunker: based on *OpenNLP*, with a model created on *Bosque* (a manually revised part of the *Floresta Sintá(c)tica* corpus):
- Dependency parser: based on *MaltParser*, also with a model created from the *Bosque* corpus;
- Nominalizer: built from the ground up, using a list of verbs and common noun suffixes, for establishing relations between verbs and nouns;
- Toponymizer: also built from the ground up, using a list of toponyms and corresponding demonyms;
- Fact extractor: built from scratch, using hand specified rules.

Many of these tools have also been used in other works in our research group, namely another Ph.D. thesis (Gonçalo Oliveira, 2013) and multiple M.Sc. theses. As for LEMPORT, in special, it has been used both inside and outside our group, even abroad, as a reference tool for lemmatization of Portuguese texts, as is the case, at least, of the following works: Gonçalo Oliveira et al. (2010), Gonçalo Oliveira et al. (2011), Gonçalo Oliveira et al. (2014), Costa et al. (2015), Gonçalo Oliveira et al. (2016), Missier et al. (2016), Hachaj and Ogiela (2016), and Wijaya and Mitchell (2016).

RAPPORTÁGICO, a system closely tied to RAPPORT, embodying one of its first versions, is also a meaningful contribution, related with the retrieval of information from Wikipedia, selecting articles instead of presenting passages as answers.

Finally, the work done in the course of this thesis has been presented in scientific papers, published in international conferences such as EPIA (Diéguez et al. (2011), Rodrigues and Gomes (2015) and Gonçalo Oliveira et al. (2017)), PROPOR (Rodrigues and Gomes (2016)), SLATE (Rodrigues et al. (2014)), and in two workshops, PÁGICO and ASSIN (*Avaliação de Similaridade Semântica e Inferência Textual*), and resulted also in two articles published in LinguaMÁTICA (Rodrigues et al. (2012) and Oliveira Alves et al. (2016)).

1.7 Outline

In this first chapter we have introduced the problem we are addressing, our research goals, the approach, and the main contributions of our work. The rest of this thesis is described in the following paragraphs.

With more detail, the next chapter (Chapter 2) provides an assessment of the state-of-the-art for NLP in general, where the background knowledge needed to get a grasp of the thesis contents is presented.

In Chapter 3, we narrow the scope of our work, presenting the best known and most relevant approaches to QA for Portuguese, ending up detailing the results of each, and identifying the benchmarks we have also used for testing and evaluating our approach.

Proceeding to Chapter 4, we describe RAPPORTÁGICO, based on one of the first stable iterations of the proposed approach, developed in the scope of PÁGICO. Although with modest results against human-backed systems, its description here serves the purpose of portraying the evolution of the approach, in addition to its use in another context.

Advancing to the main body of work, Chapter 5 describes and discusses the general approach used by RAPPORT, the proposed QA system. The chapter details each of the major modules and how they fit together in the general workflow. It shows also how the system was implemented, including other tools used or developed in the process, as is the case of LEMPORT, a lemmatizer built from scratch, which was extensively used in the main work and also in the work of others.

Next follows Chapter 6, focusing on the experimentation and evaluation of the developed system, defining the used benchmark and analyzing how RAPPORT and also RAPPORTÁGICO compare against the other known QA systems for Portuguese.

Finally, Chapter 7 starts by addressing the results achieved, drawing some final remarks and wrapping up a final conclusion on the work done.

Chapter 2

Background Knowledge

Natural language processing is arguably one of the most prominent and oldest fields of artificial intelligence (AI) (Russell and Norvig, 2003), being also seen as one of the hardest (Jones, 1994), due to its intrinsic characteristics. For instance, unlike formal languages, where a symbol refers just to one very specific meaning, in natural languages a symbol may account for multiple meanings, depending on the grammatical function and context where it is found, creating ambiguity and, thus, interpretation problems.

Apart from ambiguity, which amusingly and ironically has a major role in human interactions, there are many other issues that must be dealt with, such as defining dependencies between sentence constituents, coreference resolution, or named entity recognition, that are paramount for extracting information from text.

There are also supporting tasks that have to be addressed in most NLP related works, such as sentence splitting, tokenization, part-of-speech tagging, lemmatization, or phrase chunking, mostly in preparatory stages, that end up influencing the final outcome of applications that make use of them.

The work on this thesis involves most of the NLP tasks just mentioned, for which this chapter presents the key concepts, before proceeding to address NLP applications related to the scope of this thesis. It starts with the general and generic aspects of natural language processing, passing through the most common tasks associated to NLP, that lay the groundwork for any incursion in the field, addressing applications such as information retrieval and extraction, and ending up with automatic question answering and directly related concepts.

2.1 Natural Language Processing

Natural language processing (Jurafsky and Martin, 2008) combines both computer science, specifically AI, and linguistics — being related to the of field computational linguistics or even overlapping it — and aims at facilitating the interaction and information

exchange between human beings and computers by means of using natural language on that interaction process.

The processing of natural languages is something that is innate for us humans. Humans analyze and synthesize sentences, phrases, words and utterances, being able to both extract meaning out of them, and conveying information through them. NLP addresses some of the same tasks, for instance, breaking apart texts and sentences, and processing them for information — as is the case of information extraction and information retrieval (Baeza-Yates and Ribeiro-Neto, 1999), or text summarization (Jurafsky and Martin, 2008), to name a few of the applications of NLP.

NLP deals with both the understanding and translation of natural languages to formal languages, and the other way around, as a way of facilitating interaction between humans and machines. Areas of application of NLP may range from pragmatic and practical applications, such as machine translation and the aforementioned information retrieval and extraction, to related applications, such as conversational agents, that can engage in meaningful and realistic conversations with humans.

Although not directly related, we can find associations between question answering and the Turing test (Feigenbaum, 2003), if we can imagine that a way, at least partially, for a computer to impersonate or imitate humans (Turing, 1950) is the ability to interpret and process information represented by natural languages. That is what is typically sought with the use of conversational agents for interaction with persons (creating the illusion of communication between two humans).

NLP frequently encompasses diverse and distinct tasks regarding morphological analysis of words, syntactic and semantic analysis of texts, lexical analysis, discourse analysis, and phonetic analysis, just to mention some of the tasks. In the next sections, it is given special attention to some of the most frequent concepts and tasks used in language processing, namely those that are more relevant in the QA area, building up to what can be considered a system possessing the most frequent features of a classical QA system.

We also make a distinction between NLP tasks and applications. On the one hand, we consider tasks to be simpler programs that address a specific function, that may be auto-contained or that depend linearly from the outcome of other programs and whose output itself is of limited use for a human. On the other hand, we think of applications to be inherently more complex programs, whose processing, in addition to grouping and coordinating functions and tasks, provides a significant outcome to humans, enabling the user to perform better on decision making processes. By making this distinction, we just intend to separate tasks that may be considered auto-contained from applications that rely on prior processing or on a pipeline composed of tasks, making no claims about a strict border between tasks and applications.

2.2 Natural Language Processing Tasks

This section presents tasks that are performed in any major NLP application, question answering included. As much as possible, the tasks are introduced in the same sequence they are normally used, with the output of precedent tasks serving as the input of subsequent tasks.

2.2.1 Tokenization

More often than not, one of the first tasks in the normalization of a text is tokenization, be it sentence tokenization or word tokenization, breaking whole texts into smaller tokens easier to manipulate and that can be subjected to further analysis.

Sentence Tokenization

Sentence tokenization has as its traditional role the recognition of sentence boundaries (Grefenstette and Tapanainen, 1994), grouping sets of words in that process. It relies mainly on the identification of line breaks and punctuation, such as exclamation points and question marks, as well as periods, which, as a matter of fact, can be a highly ambiguous punctuation mark, as they are used also in abbreviations and numbers, additionally to their role as sentence boundaries.

Although sentence tokenization may appear to be a simple task, it should avoid common pitfalls such as the use of periods in the aforementioned abbreviations and numbers, that should be taken as whole words (including the period) and not as the end of a sentence, or decide whether a line break marks the end of a sentence or ir just splits a sentence in two lines. It must be also language oriented, as conventions may change across languages and locations, such as the case of using a period or a comma as a decimal mark in different languages.

Please refer to Fig. 2.1 for an example[1] of the splitting of a paragraph into multiple sentences. Other names for sentence tokenization include: sentence identification, sentence detection, or, more commonly, sentence splitting.

[1] As the work in this thesis focuses on the Portuguese language, we will use, whenever possible, Portuguese text examples, and also try to use always the same examples. The excerpt used in the example originates from a news article in the July 22, 1995 edition of the *Público* Portuguese newspaper, regarding bizarre facts on the Guinness Book of Records, and loosely translated to English it reads as: "Mel Blanc, the man who lent his voice to the most famous rabbit in the world, Bugs Bunny, was allergic to carrots. In Alaska, it is illegal to look at moose from the window of an airplane. The French King Charles VI, convinced that he was made of glass, hated traveling by coach because the vibrations could shatter his noble body... For all those who like strange and unusual details, the Guinness Book of Records will be having now a spot dedicated to all these bizarre details that brighten up our odd world."

```
Mel Blanc, o homem que deu a sua voz ao coelho mais famoso do mundo, Bugs Bunny,
era alérgico a cenouras.  No Alasca, é ilegal olhar para os alces da janela de
um avião.  O rei francês Carlos VI, convencido de que era feito de vidro, odiava
viajar de coche porque as vibrações podiam estilhaçar o seu nobre corpo...  Para
todos aqueles que gostam de pormenores estranhos e fora do vulgar, o Guiness Book
of Records vai agora passar a ter um cantinho dedicado a todos estes detalhes
bizarros que dão cor ao nosso estranho mundo.

↦

Mel Blanc, o homem que deu a sua voz ao coelho mais famoso do mundo, Bugs Bunny,
era alérgico a cenouras.

No Alasca, é ilegal olhar para os alces da janela de um avião.

O rei francês Carlos VI, convencido de que era feito de vidro, odiava viajar de
coche porque as vibrações podiam estilhaçar o seu nobre corpo...

Para todos aqueles que gostam de pormenores estranhos e fora do vulgar, o
Guiness Book of Records vai agora passar a ter um cantinho dedicado a todos estes
detalhes bizarros que dão cor ao nosso estranho mundo.
```

Figure 2.1: Sentence splitting example

Word Tokenization

Word tokenization (or, quite commonly, known simply as tokenization) has the goal of dividing sentences into sequences of words and graphic symbols, such as punctuation (Grefenstette and Tapanainen, 1994). In western languages, words are defined to be strings of contiguous alphanumeric characters, usually with white spaces, line breaks, or punctuation on both sides, and may include hyphens and apostrophes but no other punctuation marks (except for the case of abbreviations).

Depending on the implementation, tokens may be grouped in order to appear as one, as is the case of names of persons or places, and other proper nouns, or broken apart, as is the case of contractions and clitics. Other issues that tokenization has to deal with include hyphenated words or large constructs, such e-mail and website addresses, or even large numbers which may include spaces and decimal marks (such as periods or commas, depending on the language convention, as already mentioned). Please refer to Fig. 2.2 for an example regarding word tokenization.

2.2.2 Part-of-Speech Tagging

Part-of-speech tagging (POS Tagging) intends to identify the syntactic role of words in a sentence, assigning each word its proper part-of-speech, according to context (Abney, 1997). It is a hard task, as a list of words with the most common associated POS tag

```
Mel Blanc, o homem que deu a sua voz ao coelho mais famoso do mundo, Bugs Bunny,
era alérgico a cenouras.

↦

Mel_Blanc | , | o | homem | que | deu | a | sua | voz | a | o | coelho | mais |
famoso | de | o | mundo | , | Bugs_Bunny | , | era | alérgico | a | cenouras | .
```

Figure 2.2: Word tokenization example

may not be enough: a word can be a noun in a sentence and a verb or an adjective in another, depending on the syntactic structure of the sentence, or even context.

Although it depends on the language and on the purpose of the approach, which may be more or less granular, words are typically assigned to one of the following parts of speech, depending on the language: *verb*, *article*, *noun*, *pronoun*, *adjective*, *adverb*, *preposition*, *conjunction*, or *interjection*. Please refer to Fig. 2.3 for an example of POS tagging. Notice that, in the example, the tags reflect Portuguese grammatical tags, specifying also variants of pronouns and of verbs, in addition to distinguishing between nouns and proper nouns.

```
Mel_Blanc | , | o | homem | que | deu | a | sua | voz | a | o | coelho | mais |
famoso | de | o | mundo | , | Bugs_Bunny | , | era | alérgico | a | cenouras | .

↦

Mel_Blanc/PROP ,/PUNC o/ART homem/NOUN que/PRON-INDP deu/V-FIN a/ART sua/PRON-DET
voz/NOUN a/PREP o/ART coelho/NOUN mais/ADV famoso/ADJ de/PREP o/ART mundo/NOUN
,/PUNC Bugs_Bunny/PROP ,/PUNC era/V-FIN alérgico/ADJ a/PREP cenouras/NOUN ./PUNC
```

Figure 2.3: Part-of-speech tagging example

POS tagging algorithms fall into one of two distinctive groups: rule-based and stochastic (Jurafsky and Martin, 2008). As its name implies, rule-based taggers use hand-defined or automatically acquired rules, either for identifying the semantic role of a known and unambiguous word, or for specifying the role of ambiguous words, based on sequences of tags and their probabilities. That is, there are words that just have one possible classification (e.g., proper nouns, articles) that are tagged outright, and others that depend on the words next to them and respective tags. One of the best known rule-based methods is the one used in the Brill tagger (Brill, 1992).

Stochastic POS taggers make use of multiple methods, such as: hidden Markov models (HMM) (Banko and Moore, 2004) used to select tag sequences, which maximize the product of word likelihood and tag sequence probability; maximum entropy (ME)

(Ratnaparkhi, 1996); or other types types of learning like decision trees (Schmid, 1994; Heeman, 1999). In all of these cases, the system uses hand annotated texts to build and train the models, and then to test them.

2.2.3 Morphological Analysis

Morphological analysis deals with the identification, analysis and description of rules applied to words (Winiwarter and Tjoa, 1993). For instance, for a given word, it may be identified its stem, lemma, and affixes (if any), as well as possible declensions or inflections, helping in the identification of other words. This way, the system just has to store a base word (or a small set of related words) and obtain all the derivatives from it (or conversely reduce all the derivatives to that base word), with obvious gains in storage and matching processes.

Two major morphological analysis tasks are stemming and lemmatization of words, mainly as a way of normalizing texts, where both techniques aim to reduce the inflectional and derivationally related forms of a word to a common base form.

Although there are cases where both techniques end up producing the same results, lemmatization, on the one hand, obtains the lemma by using morphological analysis of words, returning a dictionary base form of the word being analyzed; stemming, on the other hand, just strips off what the used algorithm perceives as being an affix in the word, independently of the resulting base form being found in a dictionary or not. For that reason, lemmatization is usually preferred over stemming, as the resulting lemmas convey more information, being more recognizable and allowing an easier identification of their role, than the stems. Nevertheless, stemming is arguably faster.

Stemming

Stemming (Flores et al., 2010) approaches usually start with a list of suffixes that are used to create word inflections, with each word of a text being checked against that list. Whenever a suffix is found in a word, it is stripped off, leaving what is to be considered the stem of the word (Lovins, 1968). Some restrictions and exceptions apply: when the word completely matches a common suffix, or when what remains from the stemming process is below a certain character length (Willet, 2006), leaving it completely unrecognizable or unsuitable for later use, the word is left untouched. Nevertheless, stemming ignores word meanings and its results may suffer of both under-stemming (failing to relate words with the same stem) and overstemming errors (presenting the same stem for unrelated words) (Savoy, 2006).

Another approach to stemming starts with a known stem, and proceeds to use a list of suffixes to generate inflected words based on that stem. Whenever a match between

these generated words and those in a text occurs, the latter are replaced by the former (Nicolai and Kondrak, 2016).

An example of stemming is shown in Fig. 2.4, where mostly gender, number and verb suffixes are remove from words.

```
Mel_Blanc | , | o | homem | que | deu | a | sua | voz | a | o | coelho | mais |
famoso | de | o | mundo | , | Bugs_Bunny | , | era | alérgico | a | cenouras | .

↦

Mel_Blanc#Mel_Blanc ,#, o#o homem#hom que#que deu#deu a#a sua#sua voz#voz
a#a o#o coelho#coelh mais#mais famoso#famos de#de o#o mundo#mund ,#,
Bugs_Bunny#Bugs_Bunny ,#, era#era alérgico#alérg a#a cenouras#cenour .#.
```

Figure 2.4: Stemming example

Lemmatization

Lemmatization (Jongejan and Dalianis, 2009), like stemming, also uses a list of affixes, but together with corresponding base forms of words found in a lexicon and associated POS tags. Most lemmatizers start by checking words against a lexicon that contains word inflections together with their dictionary form and POS tag. If a match is found, the dictionary form of the word is retrieved. If no match is found, rules are used to deal with the affixes, checking the resulting forms against the dictionary forms in the lexicon, determining whether to stop and return a matching dictionary form, or to repeat the process with another rule. An important aspect of the lemmatization process is the use of POS tags to select which rules can be applied to a given word.

An example of lemmatization can be seen in Fig. 2.5, being easily noticeable how the lemmatization output is more readable than that resulting from stemming. Nevertheless, both techniques are useful for bundling together and processing related words, facilitating word storage, retrieval and matching.

2.2.4 Named Entity Recognition

Named entity recognition (NER) (Poibeau and Kosseim, 2001; Nadeau and Sekine, 2007; Szarvas et al., 2007) is considered a subfield of information extraction (IE), whose task is to *identify* and *classify* atomic elements accordingly to a predefined set of categories. These categories can typically be ENAMEX (for proper nouns, including names of persons, locations and organizations), TIMEX (for temporal expressions such as dates and times) and NUMEX (for numerical expressions such as money and percentages), but depends on the context and languages addressed.

```
Mel_Blanc/PROP ,/PUNC o/ART homem/NOUN que/PRON-INDP deu/V-FIN a/ART sua/PRON-DET
voz/NOUN a/PREP o/ART coelho/NOUN mais/ADV famoso/ADJ de/PREP o/ART mundo/NOUN
,/PUNC Bugs_Bunny/PROP ,/PUNC era/V-FIN alérgico/ADJ a/PREP cenouras/NOUN ./PUNC

↦

Mel_Blanc#Mel_Blanc ,#, o#o homem#homem que#que deu#dar a#o sua#seu voz#voz
a#a o#o coelho#coelho mais#mais famoso#famoso de#de o#o mundo#mundo ,#,
Bugs_Bunny#Bugs_Bunny ,#, era#ser alérgico#alérgico a#a cenouras#cenoura .#.
```

Figure 2.5: Lemmatization example

Over the years, NER systems have evolved from simple handcrafted rule-based algorithms (which can still be used today) to autonomous machine learning techniques (Nadeau and Sekine, 2007).

Handcrafted rule-based approaches rely mostly in the use of regular expressions and gazetteers for detecting patterns in text. For instance, a list of proper nouns, and a way of identifying combinations of them or their inclusion in *n-grams*[2] with other words starting with capital letters, can provide a way of identifying persons in a text. Another example is the recognition of spatial prepositions followed by a word starting with a capital letter, which is a good indicator of that word being a place or location. The same applies for time expressions, using common patterns for dates, including a list of months, and for times.

Approaches based on machine learning include supervised learning methods such as the use of hidden Markov models, maximum entropy models, support vector machines (SVM), and conditional random fields (CRF), and resort to large corpora with examples of entities, memorizing lists of entities, and creating rules for disambiguation (Nadeau and Sekine, 2007).

There are also approaches based on semi-supervised learning, that use a number of examples for starting the learning process, in a form of bootstrapping. Those seed examples can be retrieved, for example, from gazetteers (usually automatically generated), where named entities are listed and classified with a given type (Kozareva, 2006). Finally, there are also unsupervised approaches that rely in information extracted from other resources, for instance, using WordNet, for retrieving the types of entities from *synsets* (Alfonseca and Manandhar, 2002).

Fig. 2.6 presents an example of NER in a sentence — notice how *Bugs Bunny* is also classified as a person, as it is *hard* for the system to *know* that he is a *rabbit* (besides also being a fictional character).

[2] An *n-gram* is a sub-sequence of *n* items from a given sequence. For instance, *1-grams* are isolated words, *2-grams* are pairs of words, *3-grams* are triples, and so on.

```
Mel_Blanc | , | o | homem | que | deu | a | sua | voz | a | o | coelho | mais |
famoso | de | o | mundo | , | Bugs_Bunny | , | era | alérgico | a | cenouras | .

↪

<START:person> Mel Blanc <END> , o homem que deu a sua voz a o coelho mais famoso
de o mundo , <START:person> Bugs Bunny <END>, era alérgico a cenouras .
```

Figure 2.6: Named entity recognition example

2.2.5 Syntactic Analysis

Syntactic analysis (Sager, 1967; Winiwarter, 1993) is the process of determining the grammatical structure of a text, by means of identifying nouns, adjectives, verbs, etc. Also known as parsing, the syntactic analysis identifies each of the elements (tokens) of a sentence, and creates a data structure that represents the sentence and the relation between each of the elements.

The main goal of syntactic analysis is to determine the grammatical structure of a text, moreover sentences, in accordance with a given formal grammar,[3] and it can be divided into two major types: *chunk parsing* (a shallow parsing method), and *dependency parsing* (that has been implemented using both shallow and deep parsing methods). These two parsing methods, under certain restrictions on the chunk parsing side, are strongly equivalent, being possible to generate the same sentences and get the same structure from both (Covington, 2001). However, these two methods are typically used for different purposes, as we will see next.

Chunk Parsing

Phrase chunking or chunk parsing is meant to assign a partial structure to a sentence, regarding sequences of tokens and identifying relations between those sequences (Görz and Schellenberger, 2011). Chunking can also be known as a constituency grammar, as it breaks apart sentences into constituents. Amidst other characteristics, according to and reproducing Görz and Schellenberger (2011), chunks:

- never cross constituent boundaries;
- form true compact substrings of the original sentence;
- are implemented based on elementary features of the word string, like the POS tags or word types, avoiding deep lexical or structural parameterization incorporated in their implementation;

[3] A formal grammar is a set of rules that describe which strings formed from the alphabet of a formal language are syntactically valid within that language.

- are not recursive.

For phrase chunking, typically a sequence of words (tokens) in a sentence is annotated with `IOB` tags (for *inside*, *outside*, and *beginning* of chunk). The `O` tag is applied to punctuation tokens, while the `B` and `I` tags are applied to the other token types and are further characterized with the type of chunk (phrase) the tokens are part of: `NP` for noun phrase, `VP` for verb phrase, `ADJP` for adjectival phrase, and so on. Furthermore, each chunk must start always with a `B` token, and an `I` token must always be preceded by a `B` token or another `I` token. Chunking is therefore the procedure of grouping tokens according to their role in a sentence, using for the purpose `IOB` tags, with a `B` tag identifying the start of a new chunk.

An example of the output of chunking is shown in Fig. 2.7, presenting, for each token in the sentence, the corresponding POS tag and `IOB` tag, and, at the end of the figure, the tokens grouped in chunks.

```
Mel_Blanc | , | o | homem | que | deu | a | sua | voz | a | o | coelho | mais | famoso | de | o |
mundo | , | Bugs_Bunny | , | era | alérgico | a | cenouras | .

↪

token        pos          iob
Mel_Blanc    prop         B-NP
,            punc         O
o            art          B-NP
homem        n            I-NP
que          pron-indp    B-NP
deu          v-fin        B-VP
a            art          B-NP
sua          pron-det     I-NP
voz          n            I-NP
a            prp          B-PP
o            art          B-NP
coelho       n            I-NP
mais         adv          B-ADJP
famoso       adj          I-ADJP
de           prp          B-PP
o            art          B-NP
mundo        n            I-NP
,            punc         O
Bugs_Bunny   prop         B-NP
,            punc         O
era          v-fin        B-VP
alérgico     adj          B-ADJP
a            prp          B-PP
cenouras     n            B-NP
.            punc         O

↪

[NP Mel_Blanc] , [NP o homem que] [VP deu] [NP a sua voz] [PP a] [NP o coelho] [ADJP mais famoso]
[PP de] [NP o mundo] , [NP Bugs_Bunny] , [VP era] [ADJP alérgico] [PP a] [NP cenouras] .
```

Figure 2.7: Phrase chunking example

Dependency Parsing

Another form of syntactic analysis is dependency parsing (Covington, 2001; Koo et al., 2008; McDonald et al., 2005). Fig. 2.8 depicts an example of dependency parsing, using different classifiers, namely constituents (e.g., *subject*, *verb*, or *object*). The key characteristic of dependency parsing is the relation established between the tokens or words found in a sentence, going beyond strict proximity relations between them. That is specially useful for establishing relations among phrases in a sentence.

```
Mel Blanc era alérgico a cenouras .

↦

1 Mel_Blanc subject (head:  2)
2 era root (verb) (head:  0)
3 alérgico subject predicative (head:  2)
4 a postpositioned adverbial adject (head:  3)
5 cenouras argument of preposition (head:  4)
6 .  punctuation (head:  2)
```

Figure 2.8: Dependency parsing example

Whenever a relation is established between two words, one assumes the role of *head* and the other the role of *dependent*. If a word does not depend from another word, it is called *independent* (or headless). There is only one independent word in a sentence (the *root*, usually the main verb of that sentence, with `0` as its head), and dependent words are usually modifiers, subjects, objects, or complements (Covington, 2001).

The dependency parsing process, according to Covington (2001), makes use of the following basic assumptions: *unity*, *uniqueness*, *projectivity*, *word-at-a-time operation*, *single left-right pass*, and *eagerness*. Simplifying greatly any of the existing strategies, all words in a sentence are paired together in all possible combinations, either as head-to-dependent or dependent-to-head, selecting then the pairs permitted by grammar rules and by the basic assumptions just stated.

2.2.6 Semantic Analysis

Semantic analysis (Jurafsky and Martin, 2008; Hirst, 1988) deals with the meanings of words, expressions, phrases and sentences. The meaning of phrases and sentences depends on the words that constitute them, and also on the order of the words and the relations among the words on the syntactic level. Semantic analysis is also important to eliminate, when possible, ambiguities that may arise on the semantic level.

For instance, in the examples used previously, we can infer that Mel Blanc was aller-

gic to a given vegetable (a carrot is a *hyponym* of vegetable) and that he did not really lend is voice to Bugs Bunny (but was obviously the voice actor that played the rabbit).

Semantic analysis is also involved in the passage from natural languages to formal languages, making possible the building of parse trees or first order logic predicates and their analysis by computer programs. An eventual representation of Mel Blanc being allergic a carrots, a specific vegetable, is depicted in Fig. 2.9.

```
Mel Blanc era alérgico a cenouras (um vegetal).

↦

Mel_Blanc λ(x) (allergic_to x carrots) ∧ carrots λ(y) (is_a y vegetable)
```

Figure 2.9: Logic representation of a sentence example

2.2.7 Word Sense Disambiguation

Defining word sense as a commonly accepted meaning of a given word (Navigli, 2009), word sense disambiguation (WSD) involves the association of a specific word in a text or discourse with a definition or meaning (sense) which is distinguishable from other meanings potentially attributable to that word (Ide and Véronis, 1998). That is, WSD is responsible for selecting, when a word has multiple meanings, which of those meanings should be considered.

WSD depends much on external knowledge sources, such as lexica and encyclopedias, and context, including information conveyed by the sentence structure where the words are found. As such, and regarding context, the morphological classification of a word significantly improves its disambiguation — for instance, using POS tagging, for determining whether the word refers to a noun or to a verb, and choosing its meaning and eventual synonyms.

However, the most difficult task in WSD is to disambiguate between polysemic words, and, to a lesser degree, homonyms with the same part-of-speech classification. In such cases, multiple approaches exist, being the most simple of them to use the sense which is more common (independently of context), selecting which of the meanings (for instance, a synonym) has a higher frequency in a given corpus. Please refer to Fig. 2.10 for an example of WSD.

2.2.8 Anaphora and Coreference Resolution

Anaphora is a cohesion mechanism in sentences, in which an item points back to some previous item, in the same or another sentence. That is, citing Mitkov (2002), “the

```
Mel Blanc, o homem que deu a sua voz a o coelho mais famoso de o mundo, Bugs
Bunny, era alérgico a cenouras .

deu|dar

↦

dar [to give], literarily
∨
emprestar [to lend], figuratively (and the correct meaning in the example
sentence which has been used)
```

Figure 2.10: Word sense disambiguation example

interpretation of some element in the discourse is dependent on that of another and involves the use of abbreviated or alternative linguistic forms which can be recognized and understood by the hearer or the reader, and which refer to or replace previously mentioned items in the spoken or written text."

It is worth noticing that there is a close relation between anaphora and coreference resolution, to the point of one being mistaken by the other. However, a distinction is that an anaphora is "the linguistic phenomenon of pointing back to a previously mentioned item in the text as opposed to coreference, which is the act of referring to the same referent in the real world" (Mitkov et al., 2000), as in mentioning a person earlier by its name, and then later by a personal pronoun (anaphora) or by the occupation of that person (coreference).

For multiple reasons, including style, avoiding constant repetitions, or discourse economy, many items, such as persons, places, or objects, are only fully addressed once in a sentence, paragraph or even a whole document, being replaced in later references by, for example, personal pronouns (pronominal anaphora). Even in this document, concepts are initially introduced together with their initials, and later are sometimes referred using just the initials. As such, it is of paramount importance for the correct interpretation of a text that the references between any entities and corresponding replacements are properly identified. Algorithms used for addressing this problem typically start by looking first for the nearest entity that might be compatible or replaced by the referring expression (Kibble, 1997).

Please refer to Fig. 2.11 for an example where the `ele` (he) personal pronoun is replaced by the name of the person it is referring (anaphora). In the example, the same person is referred at times by its full name or just by its last name (coreference).[4]

[4]The excerpt used in the example originates from a column by Bill Gates found in the July 22, 1995 edition of the *Público* Portuguese newspaper (via The New York Times Syndicate), and, loosely translated to English, it reads as: "If IBM and other major computer makers had recognized the importance of the personal computer a few years ago, they could have adapted better to the increasingly smaller role

Se a IBM e outros fabricantes de grandes computadores tivessem reconhecido a importância do computador pessoal uns anos antes, poderiam ter-se adaptado melhor ao papel cada vez menor do «mainframe». Em vez disso, foi pouca sorte a de John Akers estar à frente da IBM quando se tornou absolutamente evidente que a empresa se deixara ultrapassar pela mudança estrutural. Akers era um extraordinário presidente executivo, muito profissional. Se ele tivesse estado à frente da IBM numa fase de estabilidade, hoje só suscitaria referências laudatórias. Em vez disso, ele demitiu-se abruptamente em 1993, em parte para marcar bem o quanto a empresa precisava de mudar. Eu admiro o modo como ele pôs os interesses da empresa acima dos seus próprios interesses.

↦

Se a **IBM** e outros fabricantes de grandes computadores tivessem reconhecido a importância do computador pessoal uns anos antes, poderiam ter-se adaptado melhor ao papel cada vez menor do «mainframe». Em vez disso, foi pouca sorte a de **John Akers** estar à frente da **IBM** quando se tornou absolutamente evidente que a **[IBM]** se deixara ultrapassar pela mudança estrutural. **[John] Akers** era um extraordinário presidente executivo, muito profissional. Se **[John Akers]** tivesse estado à frente da **IBM** numa fase de estabilidade, hoje só suscitaria referências laudatórias. Em vez disso, **[John Akers]** demitiu-se abruptamente em 1993, em parte para marcar bem o quanto a **[IBM]** precisava de mudar. Eu admiro o modo como **[John Akers]** pôs os interesses da **[IBM]** acima dos seus próprios interesses.

Figure 2.11: Anaphora and coreference resolution example

There are multiple types of anaphora, including lexical noun, noun, verb, adverb, and zero anaphora, additionally to the already mentioned pronominal anaphora.

2.2.9 Ontologies

Ontologies are an important component of information systems and information processing, including natural language processing. A well known definition states that an ontology is an explicit specification of a conceptualization (Gruber, 1995). That is, an ontology deals with the study and classification of things into predefined types, constituting an approach for knowledge representation that is capable of expressing a set of entities and their relationships, constraints, axioms and the vocabulary of a given domain (Drumond and Girardi, 2008), conveying a way to address *meaning* of *terms*, required for processing information (Estival et al., 2004).

Ontologies may have many applications. In the specific domain of NLP, ontologies are specially useful for storing information in a structured way and for supporting in-

of the «mainframe». Instead, it was bad luck for John Akers to be the head of IBM when it became absolutely clear that the company was overwhelmed by structural change. Akers was an extraordinary, very professional CEO. If he had been heading IBM in a phase of stability, today he would receive only laudatory references. Instead, he resigned abruptly in 1993, partly to mark how much the company needed to change. I admire the way he put the company's interests above his own interests."

formation retrieval (Soergel, 1999), as they:

- provide knowledge-based support of end-user searching;
- support hierarchically expanded searching;
- support well-structured displays of search results;
- provide a tool for indexing.

Ontologies usually deal with classification tasks and are a combination of multiple constituents, of which the most important are (Khoo and Na, 2006):

- **classes**: collections, concepts or types of objects;
- **relations**: how classes and individuals can be related to one another;
- **instances**: a specific concretization of a class;
- **axioms** and **rules**: statements that describe the logical inferences that can be derived from an assertion, describing the domain of application of an ontology.

Specifically regarding NLP, there are six important semantic relations that ontologies must implement (Khoo and Na, 2006), which are also important in the transposition of natural languages to formal languages:

- **hyponymy-hypernymy** — which goes commonly under the relation name `is-a` (hyponym refers to the narrower term/concept and hypernym is the broader term/concept);
- **troponymy** — which refers to broader-narrower relations between verbs;
- **meronymy** — a relation between a concept/entity and its constituent parts;
- **synonymy** — a relation where both constituents have the same or related meanings, as is the case of sense-synonyms (terms which share one or more senses), near-synonyms (which have no identical senses but are close in meaning), and partial synonyms (which share some senses but differ in some aspect);
- **antonymy** — a relation where both constituents have opposite meanings;
- **cause-effect** — a relation where a constituent results from the other (the cause can be classified as necessary or sufficient).

Ontologies are commonly encoded using ontology languages, which are expected to comply with three important aspects: conceptualization, vocabulary, and axiomatization. Ontologies are also expected to follow three requirements: extensibility, visibility, and inferenceability (Ding et al., 2005).

2.3 Natural Language Processing Applications

This section presents natural language processing applications that rely on the tasks previously described, including question answering and other applications that bear some relation or share characteristics with it.

2.3.1 Information Retrieval and Extraction

Most of human knowledge, represented in any form of natural language, but mostly in written form, is primarily unstructured information, even if for humans it has some kind of apparent structure. The exceptions include dictionaries and encyclopedias.

Turning that unstructured information into structured information is what information retrieval and information extraction are used for, providing tools and techniques for processing natural language sources. Examples of structured information are, among others, databases, ontologies, and triple stores, that are often used for storing the output of both information retrieval and extraction.

It is in this context that information retrieval and information extraction are important and are briefly described next.

Information Retrieval

Information retrieval (IR) is the task of searching for information within documents that relate to a specific user query. It deals with the representation, storage and organization of information items, and also with the access to them in a straightforward manner (Baeza-Yates and Ribeiro-Neto, 1999). The documents may be structured, semi-structured on unstructured, although when dealing with NLP most of the times we are considering unstructured documents that have to be processed in order to retrieve the needed information.

The earliest information retrieval systems are contemporary to the first computers in the late 1940s, using them to query index data records (Sanderson and Croft, 2012). Most data retrieval systems, much like text search engines, determine, in a given set of documents, which ones best match the keywords specified in the user query. However, depending exclusively on keyword matching (using the user query) may be insufficient to present the required information to the user. An evidence of this is the well known tip for using search engines, that states that a good query should contain keywords that are expected in the answer (even if not in the question).

What happens with database management systems is also similar: the user only gets data related to the query presented to the system, by pattern matching and definition of restrictions. A way of circumventing this limitation is to actively address the user

query, expanding it, selecting which terms are relevant and ignoring those that are not, as is the case of too common words (stop words) and punctuation.

Currently, IR systems deal with more than just data matching — for instance, the gathering of information about a given topic or subject. Typically, information retrieval systems identify several documents in a collection that match the user query, assigning them specific scores in function of their relevancy (Singhal, 2001).

IR systems have to analyze documents, extract syntactic and semantic information from texts, and use that information to present to the user the information requested. In this process, a key aspect is to decide the relevance of the documents — ranking them — according to the query made by the user, avoiding presenting irrelevant documents (Baeza-Yates and Ribeiro-Neto, 1999).

Information Extraction

Information extraction (IE) is the process of getting structured data from unstructured information in the text (Jurafsky and Martin, 2008). It consists of "identifying a predefined set of concepts in a specific domain, ignoring other irrelevant information, where a domain consists of a corpus of texts together with a clearly specified information need" (Piskorski and Yangarber, 2013).

Moreover, it also encompasses of the identification of instances of a particular class of relationships in a natural language text, and the extraction of relevant arguments for that relationships (Grishman, 1997). For such task, semantic classification of information elements must be performed, but, even so, it remains a shallow form of text comprehension (Moens, 2006).

Perhaps, one of the best examples of information extraction tasks, already described in Subsection 2.2.4, is named entity recognition. Coreference resolution (mentioned earlier in Subsection 2.2.8) is also considered a classic information extraction task, as is relation extraction (the detection of relationships between entities in a text), and event extraction (the identification of elements in a text that answer *who did what to whom*, *when*, *where, how*, and *why* (Piskorski and Yangarber, 2013)).

2.3.2 Open Information Extraction

Open information extraction (OIE) has obvious connections with information extraction. There is, however, an important difference in the process. If in information extraction, namely in relation extraction, we begin with a predefined set of options for the classification of entities or the relations among them (for instance, searching a text for birthdates of people, events, or locations of places), in OIE the relations are not predefined, being identified as the text is being processed (Fader et al., 2011).

According to Gamallo (2014), OIE systems can be classified into one of four different categories, in accordance with the techniques used:

- **Training data and shallow syntax** — examples of this category are TextRunner (Banko and Etzioni, 2008), the first IOE system, that uses training data, ReVerb (Fader et al., 2011) and R2A2 (Etzioni et al., 2011), both evolutions of the just mentioned TextRunner, that also use shallow syntactic analysis, and WOE^{pos} (Wu and Weld, 2010), which uses a classifier trained with a corpus automatically obtained from Wikipedia;

- **Training data and dependency parsing** — WOE^{dep} (Wu and Weld, 2010) and OLLIE (Mausam et al., 2012) are examples of this category, using training data represented by means of dependency trees;

- **Rule-based and shallow syntax** — these systems resort to lexical-syntactic patterns hand-crafted from POS tagged text, having as examples ExtrHech (Zhila and Gelbukh, 2013), and LSOE (Xavier et al., 2013);

- **Rule-based and dependency parsing** — systems in this category make use of hand-crafted heuristics operating on dependency parse trees, such as ClauseIE (del Corro and Gemulla, 2013), CSD-IE (Bast and Haussmann, 2013), KrakeN (Akbik and Löser, 2012), and DepOE (Gamallo et al., 2012).

2.3.3 Question Answering

Question answering is the process of automatically answering a question formulated in natural language, using also natural language. It is usually classified as a subfield of IR, aiming at retrieving an answer with precise information about a question, rather than a collection of documents. As in other subfields of IR, there is a need for semantic classification of entities, relations between entities, and of semantically relevant phrases, sentences, or even larger excerpts (Moens, 2006).

QA systems can be classified accordingly to multiple characteristics. Observing Table 2.1 (adapted from Maybury (2003)), QA systems can be divided in three types of question/answer complexity (easy, moderate and hard) in accordance with: the volume and quality of the source texts, the kind and type of corpora used, and how difficult it is to generate answers.

QA systems are also characterized on the type of user query — **form** (*keywords*, *phrases* and *questions*), **type** (*who*, *what*, *when*, *where*, *how*, *why* and *what-if*) and **intent** (*request*, *command* and *inform*) — and on the type of answers provided — *named entities*, *phrases*, *factoids*, *links to documents* and *summaries*.

Table 2.1: QA systems classified on complexity (Maybury, 2003)

Question, Answer Complexity	Source Volume, Quality	Corpus, Resource Model	Answer Integration & Generation
Moderate Q., Easy A.	Small (100s MB), Static, High quality source	Encyclopedias, Technical manuals	Easy
Easy to Moderate Q., Moderate A.	Small to high (10 GB), Dynamic, Variable quality sources	Web	Moderate
Hard Q., Hard A.	Very high, Real-time, Streaming, Dynamic variable quality sources	Varied, Multilingual (in Answer and in Question)	Hard

What is expected of these systems is simple: the user types a question in natural language and the system produces the correct answer. The usual steps include:

1. analyzing the question posted by the user;
2. gathering information in the documents that may relate to the question;
3. processing the answer(s);
4. presenting the answer(s) to the user.

QA may use two distinct methods for analyzing textual elements: shallow and deep (Bos and Markert, 2005). Shallow semantic techniques depend greatly on tokenization and lemmatization, and then on the overlapping of terms between question and possible answers. However, these terms are processed in such ways that it is possible to assess its importance — for instance, through the use of document frequency. Deep semantic techniques depend on the semantic interpretation of texts through the use of, for example, some kind of ontology.

Regarding scope, QA can be divided in two categories: open domain and closed domain. On the one hand, open domain QA deals with questions whose answer can be found in a collection of natural language texts. The target of the question does not have to be a specific one, being allowed multiple topics and contexts. On the other hand, closed domain QA usually works in relation to a specific context and well defined structure, which improves the interpretation of the questions and the answers provided. Alas, most of time, it is the context or the lack of it that improves or worsens the performance of a QA system.

QA may operate on structured data, such as a database or an ontology, or on unstructured data, as is the case of text. The analysis of structured data may be easier, coming from the supervised processing of unstructured data, which may lead to a more limited scope of the retrieved information, although it can be more accurate. Nevertheless, unstructured data is what we get from the vast majority of all human produced documents, although it is arguably more difficult to process (by machines, that is).

A QA system may return answers in the form of text snippets (passages) or generate its own answers. Usually, systems that use unstructured data provide the answers in the form of text snippets or excerpts. Systems that rely on structured data try to compose answers putting together the various elements found to be part of an expected answer (Schlaefer et al., 2006).

Most QA approaches function as described above or rely in some variation of it as default. However, QA has incorporated new techniques over time, such as annotation of sentences that can be used as answers, or the generation of questions that are then used at run-time for the user to choose one of them. Next, we present a brief presentation of common techniques.

Question Oriented QA

Question oriented QA approaches rely essentially on the extraction of information from the question in order to better classify it and hence to retrieve an adequate answer. This is one of the first techniques used in QA.

Some of those approaches depend on the use of *question targets* (or *Qtargets*, as answer types are called in Hermjakob (2001)). The reasoning is that each type of question has a specific type of answer, as we can see in the examples shown in Fig. 2.12 (reproduced from Hermjakob (2001)).

```
Question:  How long would it take to get to Mars?
Qtarget:   temporal-quantity;

Question:  When did Ferraro run for vice president?
Qtarget:   date, temp-loc-with-year; =temp-loc;

Question:  Who made the first airplane?
Qtarget:   proper-person, proper-company; =proper-organization;

Question:  Who was George Washington?
Qtarget:   why-famous-person;

Question:  Name the second tallest peak in Europe.
Qtarget:   proper-mountain;
```

Figure 2.12: Question type examples (Hermjakob, 2001)

Other works may use different terminology or types, but the basis is usually the same. After classifying the questions and determining their types, the system proceeds to parse the answer candidates (answers with the expected question type), making use of named entity recognition and ontologies. After that step, parse trees of the question and the candidate answers are created and compared, after which an answer is selected and proposed.

Another type of approaches that focus on the questions are those that reformulate the user queries in order to improve the matching probabilities with the expected answers, as we can find in Brill et al. (2002). This procedure can be observed in Fig. 2.13 (reproduced also from Brill et al. (2002)).

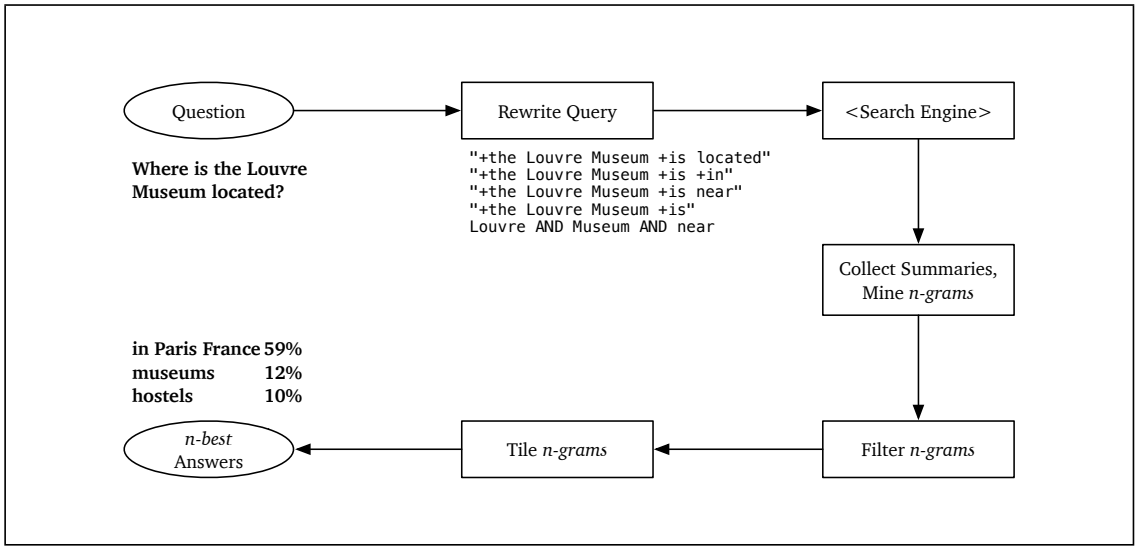

Figure 2.13: Query reformulation example (Brill et al., 2002)

The main steps of such systems are: query reformulation, *n-gram* mining, *n-gram* filtering and *n-gram* tiling. In that figure, it is easy to observe the basic operation of that approach. Each query (question) is rewritten according to its type, or, to be more precise, question pattern, in order to match the most common answer beginnings. Then each of the *n-grams* is used on a search engine for retrieving sentences or summaries with that same *n-grams*. Then the probable answer *n-grams* are filtered and recombined to produce the final answer.

QA by Virtual Annotation

Question answering by virtual annotation (Prager et al., 2006) involves the use of annotation to better describe the sentences (and possible answers) of a document. Those annotations are made offline and mostly at the word (token) level, and are then used to simplify the matching between the user query and the wanted answer.

Some of the approaches that use this technique are those of Prager et al. (2000a,b, 2001). These approaches have similarities with that described in Hermjakob (2001), however they focus more on the analysis of the answers than of the questions.

In this kind of approaches, tokens in a sentence are annotated with special tags (*QA-Tokens* in the works of Prager et al. (2000a,b, 2001)) that denote the type of the tokens (much like in named entity recognition), and that also correspond to a specific type of questions. The matching between questions and sentences (answers) is performed by

means of searches in indices with the special tags and associated sentences. A list of the tags can be seen in Table 2.2 (reproduced from Prager et al. (2000b)), together with the associated question types and examples.

Table 2.2: Predictive annotation QA tags (Prager et al., 2000b)

QA-Token	Question type	Example
PLACE$	Where	In the Rocky Mountains
COUNTRY$	Where/What country	United Kingdom
STATE$	Where/What state	Massachusetts
PERSON$	Who	Albert Einstein
ROLE$	Who	Doctor
NAME$	Who/What/Which	The Shakespeare Festival
ORG$	Who/What	The US Post Office
DURATION$	How long	For 5 centuries
AGE$	How old	30 years old
YEAR$	When/What year	1999
TIME$	When	In the afternoon
DATE$	When/What date	July 4th, 1776
VOLUME$	How big	3 gallons
AREA$	How big	4 square inches
LENGTH$	How big/long/high	3 miles
WEIGHT$	How big/heavy	25 tons
NUMBER$	How many	1,234.5
METHOD$	How	By rubbing
RATE$	How much	50 per cent
MONEY$	How much	4 million dollars

An example of this type of annotation in a sentence can bee seen in Fig. 2.14 (reproduced again from Prager et al. (2006)), with the *meaningful* tokens surrounded by tags that classify them. The tokens can also be encapsulated for better characterizing them, specifying a more general classification and then a more specific one.

```
The <THING>specimens</THING> were sent in <DATE><YEAR>1932</YEAR></DATE>
by <PERSON>Howard Carter</PERSON>, <ROLE>the British archaeologist</ROLE>
who discovered <PERSON>Tutankhamun</PERSON>'s treasure-laden tomb in the
<PLACE>Valley of the Kings</PLACE> at <PLACE><CITY>Luxor, Egypt</CITY></PLACE>,
in <DATE><YEAR>1922</YEAR></DATE>.
```

Figure 2.14: Passage annotation example (Prager et al., 2006)

These categories will correspond to interrogative pronouns such as *who*, *where*, *when*, *how much* or *how long*. Then the sentences can be identified in the text by simple pattern-matching techniques (Prager et al., 2000b) using the tags appended to the words. For instance, the question "Who wrote Hamlet?" can be rewritten as "**PERSON$** write Hamlet" (after lemmatization). Then the system "just" has to search for a sentence with the same tags, in addition to the remaining words, and send it to the user (Prager et al., 2001).

Predictive Questioning QA

Predictive questioning QA approaches are those which rely on the generation of questions based on the available corpus, to assist the user in the creation of questions or in the selection of questions proposed by the system — see, for instance, Harabagiu et al. (2005); Hickl et al. (2006a,b, 2007) — associated with the sentences or excerpts that served for predicting questions.

The foundations for such approaches lay on the creation of question-answer pairs, in the guise of a frequently asked questions (FAQ) system, suggesting multiple matches for a user question or presenting related questions to those made by the user, guiding him in the process of selecting further questions on a given topic.

These systems start by annotating the sentences in an offline stage, and classifying each of their words with the appropriate terms, including named entity recognition, identifying the major topics or aspects in a text. Then, based on that classification and identified topics of a text, questions are generated. Those questions are then stored in a database (called a question-answer database, QUAB, by Harabagiu et al. (2005)), associated with the respective answers for future use.

When the user posts a question to the system, that question is compared, using various similarity metrics (from *tf-idf* to the use of semantic information (Harabagiu et al., 2005; Hickl and Harabagiu, 2006)), with the questions already found in the database. The questions in the database are then ranked and presented to the user in order for the user to choose one or more of them (or even ignoring them) and access the respective answers.

In Fig. 2.15 (reproduced from Harabagiu et al. (2005)), it is depicted an example of a question posed to the system, its answer and also other previously generated questions that may relate to the question, that the user may select and then get the associated answer.

IR-Based Question Answering

Information retrieval based question answering (Kolomiyets and Moens, 2011) relies on the vast amounts of information available as text, such as specific corpora or even Wikipedia. In this paradigm, the system, given a user question, applies information retrieval techniques that are used to extract passages from the texts in the corpus used, based on the text of the question.

Generally, this type of approach start by processing — performing sentence splitting, tokenization, stemming or lemmatization, and so on — and indexing all documents in a given corpus, associating the documents to their contents.

The user's question is then used both for determining the expected answer type, by

(Q1) What weapons are **included in** Egypt's stockpiles?

(A1) The Israelis point to comments made by former President Anwar Sadat, who in 1970 stated that Egypt has biological weapons stored in refrigerators ready to use against Israel if need be. The program might include "**plague**, **botulism toxin**, **encephalitis virus**, **anthrax**, **Rift Valley fever** and **mycotoxicosis**."

(Q2) Where did Egypt **inherit** its first stockpiles of chemical weapons?

(Q3) Is there evidence that Egypt has **dismantled** its stockpiles of weapons?

(Q4) Where are Egypt's weapons stockpiles **located**?

(Q5) Who **oversees** Egypt's weapons stockpiles?

Figure 2.15: User question and related questions example (Harabagiu et al., 2005)

means of specific words that denote what is expected (e.g., *who*, for people; *where*, for locations; and so on), or present entities (yielding clues such as the question referring persons, locations or times, among others), and for identifying the keywords used for searching the documents index. After searching the index, passages are selected and ranked based on the contents of the query, after which the type of the question is used for further restricting the selected passages that will be ultimately presented to the user.

Question answering systems based on IR typically follow the framework shown in Fig. 2.16 (reproduced from Jurafsky and Martin (2008)), where most of the processing stages are made at run-time (except for document indexing).

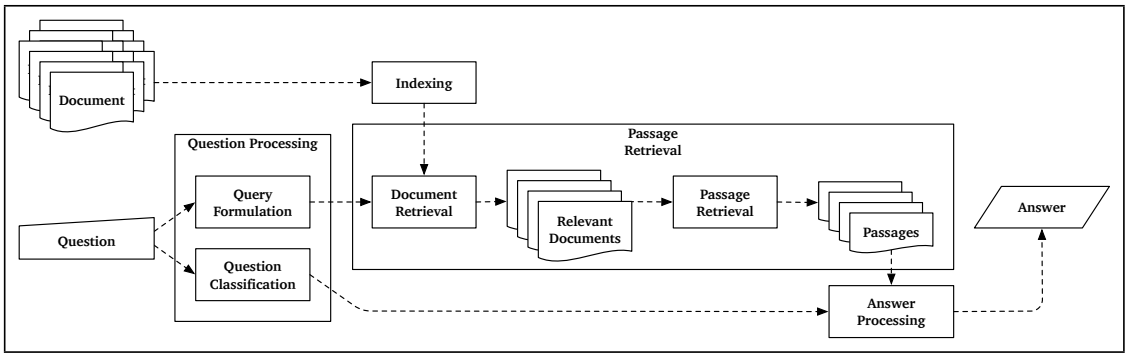

Figure 2.16: A typical framework for a IR-based QA system (Jurafsky and Martin, 2008)

Knowledge-based Question Answering

In this paradigm, the system must start by creating a knowledge-base — a formal representation of knowledge associated with inference methods — that will be used to synthesize the answers to the questions. The terms found in the documents are processed,

undergo linguistic analysis and a representation of the contents of the documents is stored in the knowledge-base (Rinaldi et al., 2003).

The system then builds a semantic representation of the query (for instance, using a full predicate calculus statement) or identifies a simple relation between a known entity and some unknown item. In either case, representations usually resort to triples, identifying a *subject*, *predicate* and *object*.

After a match is attempted between the semantic representation of the query and the knowledge-base built to represent the contents of the text, the answers are presented to the user based on the matches, and associated sentences in the documents.

Knowledge-based question answering systems, although sharing some similarities with IR based systems, tend to adopt logical representations of facts, for instance, through the use triples backed by ontologies, often implemented using RDF[5] triple stores and SPARQL[6] to query them (Unger et al., 2012), or similar data repositories. They also usually rely on semantic analysis for extracting meaning out of sentences.

[5]The Resource Description Framework (RDF) is a metadata data model used for the description or modeling of information, mainly for web resources.

[6]SPARQL (SPARQL Protocol and RDF Query Language) is a RDF query language, used to query triple stores and retrieve triples.

Chapter 3

Related Work

Now that the key concepts have been introduced, a survey on related work to question answering for the Portuguese language is presented next, focusing on both QA systems and corpora.

We start by describing the corpora, namely CHAVE and the Portuguese Wikipedia, and the treebanks used in this work. Next we present a brief description of some of the most relevant QA systems for Portuguese, dividing them accordingly to the benchmarks used to evaluate them, namely: the Conference and Labs of the Evaluation Forum (formerly known as Cross-Language Evaluation Forum) QA tracks for Portuguese (QA@CLEF), from 2004 to 2008; PÁGICO, in 2012; and unspecified evaluations.

3.1 Corpora and Treebanks

In this section we focus on the various treebanks and corpora used in our approach, both for training models for some of the tools, and for the evaluation of RAPPORT and RAPPORTÁGICO. We also address some of the formats used for storing or representing data, either for the corpora, or for use by the tools that processed the texts.

3.1.1 Corpora

For testing purposes, we have resorted to two distinct corpora: CHAVE and the Portuguese Wikipedia. The CHAVE corpus has been used in the context of the QA@CLEF tracks, for testing RAPPORT. Wikipedia has been used in both the contexts of the QA@CLEF tracks and of PÁGICO, for testing RAPPORT, in conjunction with CHAVE, and RAPPORTÁGICO.

CHAVE

For testing RAPPORT, we have used mainly the CHAVE corpus (Santos and Rocha, 2005), a collection of the 1994 and 1995 editions — a combined total of 1456 — of newspapers *Público* (Portuguese) and *Folha de São Paulo* (Brazilian). Each of the editions usually comprehends over one hundred news articles, or documents, identified by: *id*, number, date, category, author (not always present), and the text of the document itself. Newspaper *Público* is composed of 51,751 documents (164 MB) in 1994, and 55,070 documents (176 MB) in 1995. By its turn, newspaper *Folha de São Paulo* is composed of 51,875 documents (108 MB) in 1994, and 52,038 documents (106 MB) in 1995 (Magnini et al., 2007).

Each edition of the newspapers in CHAVE is found in its own file, using the SGML[1] format, and each of the documents is characterized as seen in Fig. 3.1, with its data delimited by the tags `DOC`, `DOCNO`, `DOCID`, `DATE`, `AUTHOR`, and `CATEGORY`.

```
<DOC>
<DOCNO>PUBLICO-19950722-157</DOCNO>
<DOCID>PUBLICO-19950722-157</DOCID>
<DATE>19950722</DATE>
<CATEGORY>Sociedade</CATEGORY>
[...]
O estranho mundo do Guiness Book
Mel Blanc, o homem que deu a sua voz ao coelho mais famoso do mundo, Bugs Bunny,
era alérgico a cenouras.  No Alasca, é ilegal olhar para os alces da janela de
um avião.  O rei francês Carlos VI, convencido de que era feito de vidro, odiava
viajar de coche porque as vibrações podiam estilhaçar o seu nobre corpo...  Para
todos aqueles que gostam de pormenores estranhos e fora do vulgar, o Guiness Book
of Records vai agora passar a ter um cantinho dedicado a todos estes detalhes
bizarros que dão cor os nosso estranho mundo.
[...]
</DOC>
```

Figure 3.1: A CHAVE document

Additionally to the format characteristics of the editions and of each of the news articles, it is worth pointing out that *Público* uses European Portuguese and *Folha de São Paulo*, Brazilian Portuguese, whose differences ultimately affect, even if residually, the output of the tools used to process the text. Both newspapers' texts tend to follow a journalistic writing style: written in the third person (except for interviews or some opinion articles); short and precise sentences; accessible for most of the population; factual and regarding specific events; and starting with the most important points, and then elaborating them.

[1] SGML stands for Standard Generalized Markup Language.

Portuguese Wikipedia

In addition to the CHAVE corpus, the Portuguese Wikipedia was also used, in the form of snapshots or dumps, containing articles, disambiguation pages, portals, lists, categories, and so on. Both CHAVE and a snapshot dated from November 2006 of the Portuguese Wikipedia have been used in the QA@CLEF tracks for Portuguese in 2007 and 2008; a collection of documents extracted from the Portuguese Wikipedia (dated from April 25, 2011) was used in PÁGICO.

Regarding the snapshot of Wikipedia from 2011 or, more properly, the collection of documents used in PÁGICO, it was composed of 689,629 documents (9.4 GB) extracted from a larger set, of which 32,900 were predefinition pages, 5,006 were disambiguation pages, 574,077 were redirection pages, 9,678 were media pages, and 856,005 were articles (Simões et al., 2012).

As for the snapshot of Wikipedia used in the QA@CLEF tracks for Portuguese in 2007 and 2008, it was based in a regular snapshot dated from November 2006, maintaining entries or articles of Wikipedia pages, and excluding "other types of data such as images, discussions, categories, templates, revision histories, as well as any files with user information and meta-information pages" (Forner et al., 2009), yielding a total of 8.6 GB in files.

The documents contained in the snapshots are found in the HTML[2] format, which implies that, to access their contents, they have to be either *scraped* (selecting what is useful content and what is just structure) or at least have the HTML tags removed.

3.1.2 Treebanks

Treebanks were essentially used for creating models, as it happened with some of the *Apache OpenNLP* tools, and for testing some of our tools. For instance, *Bosque* (Freitas et al., 2008) was used for testing LEMPORT and for creating models for the dependency parser used in RAPPORT, and *Floresta Virgem* (Afonso et al., 2002) was used for creating the models for the *OpenNLP* tools. Both treebanks are part of *Floresta Sintá(c)tica*.[3]

The *Floresta Sintá(c)tica* treebank was created from the output of the PALAVRAS parser (Bick, 2014) applied on: Portuguese and Brazilian newspaper texts; scientific, literary and transcribed spoken texts (also in both variants of Portuguese); and a Brazilian cultural blog.

The *Floresta Virgem* treebank is composed of next to 95,000 sentences, containing close to 1,600,000 words, retrieved from the beginning of both corpora CETENFolha

[2]HTML stands for Hypertext Markup Language.

[3]The *Floresta Sintá(c)tica* treebank is available at: `http://www.linguateca.pt/floresta/`. A detailed description and manual of *Floresta Sintá(c)tica*, including *Bosque*, can be read at: `http://www.linguateca.pt/floresta/BibliaFlorestal/completa.html` [Accessed: February 2017].

and CETEMPúblico. These corpora, like CHAVE, are made up from the newspapers *Folha de São Paulo* and *Público*, but just from 1994, in the case of the former, and from 1991 to 1998, in the case of the latter (Rocha and Santos, 2000; Santos and Rocha, 2001). We have used *Floresta Virgem* due to its apparent similarities to the CHAVE corpus and, to a lesser degree, given the style, to the Portuguese Wikipedia.

The *Bosque 8.0* treebank is the last version of a manually revised part of the *Floresta Sintá(c)tica* treebank (Afonso et al., 2002), namely a subset of *Floresta Virgem*, made available by and through Linguateca. *Bosque* contains around 120,000 tokens with annotations at various syntactic levels for the Portuguese portion of it, and around 70,000 for the Brazilian portion.

3.1.3 Formats

Two of the most used data formats for Portuguese texts are *árvores deitadas*, developed by Linguateca, and the widely used CoNLL-X. Both are described next.

Árvores Deitadas File Format

Árvores deitadas[4] is one of the formats used by Linguateca for the representation of the contents of some of its corpora, including *Bosque* and *Floresta Virgem*. *Árvores deitadas* follows the format depicted in Fig. 3.2, being in its essence a tree representation of the syntax structure of a sentence.

```
A1
STA:fcl
=SUBJ:np
==H:prop('Mel_Blanc' M S) Mel_Blanc
=P:vp
==MV:v-fin('ser' IMPF 3S IND) era
=SC:adj
==H:adj('alérgico' M S) alérgico
=PIV:pp
==H:prp('a') a
==P<:np
===H:n('cenoura' <np-idf> F P) cenouras
=.
```

Figure 3.2: *Árvores deitadas* file format

Each sentence is assigned textual information, including the sequence number of the sentence in the text and the textual contents of the sentence. Each sentence may

[4] "*Árvores deitadas*" can be loosely translated to English as "trees laid down," in reference to a sentence's tree structure that is flatten down.

have one (A1) or more analysis (A2, A3, ...), in case it may be ambiguous. As usual, the root node is the highest node in the tree, with all other nodes depending on the root node or another node. Each node has data regarding the syntactic function of the word in that node, its POS tag, lemma (with gender and number) and the current form of the word in the sentence.

CoNLL-X File Format

The CoNLL-X format has its origin in the yearly Conference on Computational Natural Language Learning (CoNLL), and is virtually a standard in file formats for use with NLP tools: *MaltParser* uses it and even the *Apache OpenNLP* toolkit uses it as one of the formats for training models. The CoNLL-X format is, as a matter of fact, an extension of the original CoNNL format, and, by its turn, has also an extension, called CoNLL-U.

The data in the CoNLL-X format adheres to the following rules (retrieved from Buchholz and Marsi (2006)), with each of the fields being described in Table 3.1 (adapted also from Buchholz and Marsi (2006)):

- All the sentences are in one text file and they are separated by a blank line after each sentence.
- A sentence consists of one or more tokens.
- Each token is represented on one line, consisting of 10 fields.
- Fields are separated from each other by a *tab*.

An example of a sentence of text in the CoNLL-X data format can be observed in Fig. 3.3, with the fields ID, FORM, CPOSTAG, POSTAG, LEMMA, HEAD and DEPREL filled.

3.2 Approaches to QA@CLEF Tracks for Portuguese

The Conference and Labs of the Evaluation Forum (CLEF)[5] is a forum that promotes research and development in multilingual information access. It aims specifically at European Languages (at times including Basque, Bulgarian, Dutch, English, French, German, Greek, Italian, Portuguese, Romanian and Spanish) and provides infrastructures and test suites for the use of system developers, being one of the most prominent workshops that addresses QA in its agenda.

Even though the QA@CLEF evaluations have started in 2003, just from 2004 onwards Portuguese has been one of the addressed languages, namely in monolingual

[5]The CLEF's web site can be found at http://www.clef-campaign.org/ [Accessed: February 2017].

Table 3.1: CoNLL-X file format reproduced from(Buchholz and Marsi, 2006)

Field number	Field name	Description
1	ID	Token counter, starting at 1 for each new sentence.
2	FORM	Word form or punctuation symbol.
3	LEMMA	Lemma or stem (depending on particular data set) of word form, or an underscore if not available.
4	CPOSTAG	Coarse-grained part-of-speech tag, where tagset depends on the language.
5	POSTAG	Fine-grained part-of-speech tag, where the tagset depends on the language, or identical to the coarse-grained part-of-speech tag if not available.
6	FEATS	Unordered set of syntactic and/or morphological features (depending on the particular language), separated by a vertical bar (|), or an underscore if not available.
7	HEAD	Head of the current token, which is either a value of ID or zero ('0'). Note that depending on the original treebank annotation, there may be multiple tokens with an ID of zero.
8	DEPREL	Dependency relation to the HEAD. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT.'
9	PHEAD	Projective head of current token, which is either a value of ID or zero ('0'), or an underscore if not available. Note that depending on the original treebank annotation, there may be multiple tokens an with ID of zero. The dependency structure resulting from the PHEAD column is guaranteed to be projective (but is not available for all languages), whereas the structures resulting from the HEAD column will be non-projective for some sentences of some languages (but is always available).
10	PDEPREL	Dependency relation to the PHEAD, or an underscore if not available. The set of dependency relations depends on the particular language. Note that depending on the original treebank annotation, the dependency relation may be meaningful or simply 'ROOT.'

evaluations, until 2008. After 2008, there were changes regarding both the corpora, first from newspapers and Wikipedia to European legislation and eventually to biomedical texts, and the type of expected answers. Additionally, after that year, Portuguese systems virtually stopped to participate in the evaluations (Peñas et al., 2010a,b).[6] As such, we have chosen to restrict the *benchmarks* to the QA@CLEF Portuguese tracks from 2004 to 2008.

Although some of the configuration details of the QA tracks for each language have changed over the years, typically they are as follow (Magnini et al., 2005; Vallin et al., 2006; Magnini et al., 2007; Giampiccolo et al., 2008; Forner et al., 2009):

- There are three types of questions: *factoid questions*, *definition questions* and *closed list questions*.

[6] At least, in the following years, there was no Portuguese participants in 2009 and just one in 2010.

```
Mel_Blanc , o homem que deu a sua voz a o coelho mais famoso de o mundo , Bugs_Bunny , era alérgico a cenouras .

↦

 1          Mel_Blanc     mel_blanc     prop          prop          _          21          SUBJ          _          _
 2          ,             ,             punc          punc          _          1           PUNC          _          _
 3          o             o             art           art           _          4           >N            _          _
 4          homem         homem         n             n             _          1           N<PRED        _          _
 5          que           que           pron          pron-indp     _          6           SUBJ          _          _
 6          deu           dar           v             v-fin         _          4           N<            _          _
 7          a             o             art           art           _          9           >N            _          _
 8          sua           seu           pron          pron-det      _          9           >N            _          _
 9          voz           voz           n             n             _          6           ACC           _          _
10          a             a             prp           prp           _          6           PIV           _          _
11          o             o             art           art           _          12          >N            _          _
12          coelho        coelho        n             n             _          10          P<            _          _
13          mais          mais          adv           adv           _          12          N<            _          _
14          famoso        famoso        adj           adj           _          13          H             _          _
15          de            de            prp           prp           _          12          N<            _          _
16          o             o             art           art           _          17          >N            _          _
17          mundo         mundo         n             n             _          15          P<            _          _
18          ,             ,             punc          punc          _          12          PUNC          _          _
19          Bugs_Bunny    bugs_bunny    prop          prop          _          12          APP           _          _
20          ,             ,             punc          punc          _          12          PUNC          _          _
21          era           ser           v             v-fin         _          0           ROOT          _          _
22          alérgico      alérgico      adj           adj           _          21          SC            _          _
23          a             a             prp           prp           _          22          A<            _          _
24          cenouras      cenoura       n             n             _          23          P<            _          _
25          .             .             punc          punc          _          21          PUNC          _          _
```

Figure 3.3: An example of text in the CoNLL-X format

- There are 200 questions to be answered in each of the languages.
- The texts containing the answers are SGML tagged, with each document having a unique identifier that competing systems have to return along with the answer, in order to support it.
- The corpora are large, unstructured, open-domain text collections, made up of from newspapers and news agency articles, with the Wikipedia version of each language also used as a corpus in 2007 and 2008.
- The 200 questions given as input in the tasks are mostly related to entities, having some questions that are definition questions and others that do not have any answer in the corpora, being expected a nil answer (around 10% in both cases).
- The answers, in addition to a text snippet, should provide the document identifier that supports it, and then be judged by human assessors as *right* or *wrong*; *unsupported*, if the *docid* does not support it; or *inexact*, if the answer contains more or less information than required (Vallin et al., 2006).
- Regarding the number of accepted answers, it was possible to present multiple answers for each question, although just the first answer would be considered Magnini et al. (2007) — this, however, had some exceptions, as in 2006, where there were also results contemplating up to ten of all submitted answers.

For Portuguese, the corpora used have been CHAVE (the Portuguese newspaper *Público* and the Brazilian *Folha de São Paulo*, from 1994 and 1995), and later, the Portuguese Wikipedia, as a complementary corpus to CHAVE.

An XML example of a question used in CLEF and its accepted answer can be seen in Fig. 3.4. Notice that the answer does not only present the passage, but also where it can be found (the identifier of the news article, composed of the newspaper name, the edition date and the article number within the edition).

```
<pergunta ano="2004" id_org="1123" categoria="F" tipo="OBJECT" restrição="X"
ling_orig="PT" tarefa_pt="0173" tarefa_de="0195" tarefa_it="0191">
  <texto>A que era alérgico Mel Blanc?</texto>
  <resposta n="1" docid="PUBLICO-19950722-157">cenouras</resposta>
</pergunta>
```

Figure 3.4: A QA@CLEF question from 2004

Next, we present the Portuguese systems that competed in the QA@CLEF tracks for Portuguese, providing a brief description of each of them, alongside their general architecture.

3.2.1 Senso

The Senso Question Answering System (Saias and Quaresma, 2007, 2008) — formerly referred to as PTUE in some of QA@CLEF tracks (Quaresma et al., 2004; Quaresma and Rodrigues, 2005a), and affiliated with the Department of Computer Science of the University of Évora[7] — uses a local knowledge database, providing semantic information for text search terms expansion.

It is composed of five major modules: *query*, *libs*, *ontology*, *solver*, and *web interface*, which can be described as follows:

- The *query* module performs the question analysis, creating a search query, and selects a set of relevant documents for each question.
- The *libs* module manages the corpora, whose texts are stemmed and indexed using an information retrieval library.
- The *ontology* module (also called *local knowledge base* module by the authors) has a starting knowledge base, containing common sense facts about places, entities and events, which is available to a logic tool with inference capabilities, that helps the automatic capture of sentence meaning.
- The *solver* module performs a search for candidate answers using both a *logic solver*, that looks for answers to a question, using data in the local knowledge base, and an *ad-hoc solver*, which uses case-based answer detection for questions

[7] The institution's web site can be found at `http://www.di.uevora.pt/` [Accessed: February 2017].

where the answer can be directly detected in the texts of the corpora. The results of both submodules are merged into a weighted sorted list of candidate answers, which is processed, filtered, adjusted and reordered by an answer validator.

- Finally, the *web interface* module serves the sole purpose of providing an easy interface to the system.

Between the first incarnation of Senso and its last ones, major developments have been made to improve the information retrieval system (contained in *libs* module) and to the ontology (Quaresma et al., 2004; Saias and Quaresma, 2007).

A visual representation of these modules and of how they are connected can be seen in Fig. 3.5, reproduced from Saias and Quaresma (2008).

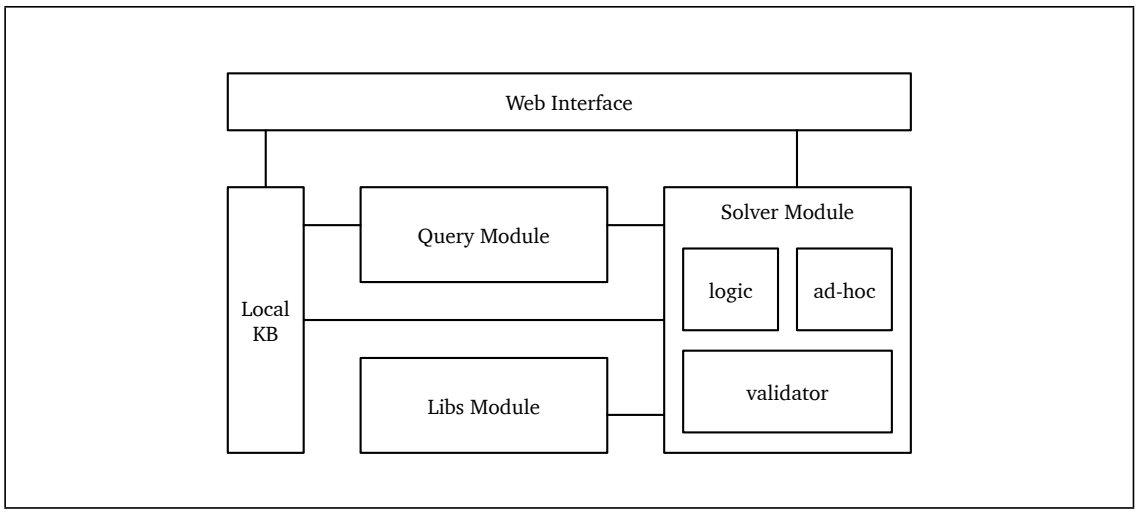

Figure 3.5: Modules and structure of Senso (Saias and Quaresma, 2008)

3.2.2 Esfinge

Esfinge is a general domain Portuguese question answering system that tries to take advantage of the large amount of information found in the Web (Costa, 2006b,a,c, 2008). Esfinge is close to more traditional QA approaches, relying mostly on pattern identification and matching, and is composed by six different modules, in the following order: *question reformulation*, *passage extraction*, *n-gram harvesting*, *NER in the n-grams*, *n-gram filtering* and *search for long answers*.

Each question is analyzed and a tentative answer is created. For instance, a probable answer for a "What is *X*?" question will most probably start with a "*X* is ..." pattern. Then this probable answer beginning is used to search the Web (or other indexed corpora), through the use of text search engines, in order to find possible answers that begin with the same pattern. In the following stages of the process, *n-grams* are scored

and NER is performed in order to improve the performance of the system. Finally, each passage is associated to the sentence that contains it, being possible to access the sentences through the passages. Esfinge is affiliated with Linguateca.[8]

A visual representation of the structure of Esfinge can be seen in Fig. 3.6, reproduced from Costa (2006b).

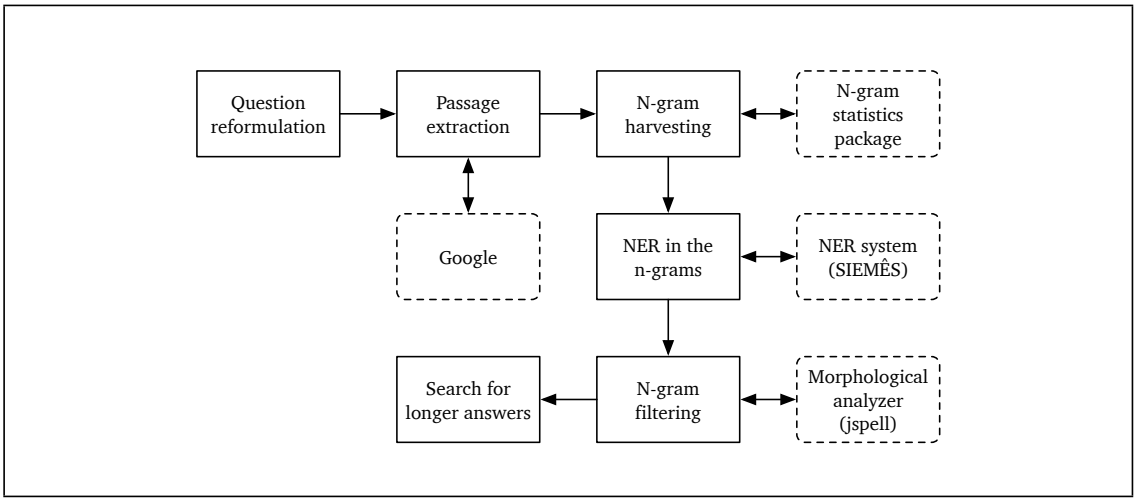

Figure 3.6: Structure of Esfinge (Costa, 2006b)

3.2.3 RAPOSA

The RAPOSA (FOX) Question Answering System is a QA system for Portuguese that tries to provide a continuous online processing chain from question to answer, combining the stages of information extraction and retrieval (Sarmento, 2006; Sarmento and Oliveira, 2007; Sarmento et al., 2008a,b). The system involves expanding queries for event-related or action-related factoid questions using a verb thesaurus, automatically generated through the use of information extracted from large corpora.

RAPOSA consists of six modules more or less typical on QA systems, namely:

- a *question parser*, for identifying the type of the question, the type of the answer, its arguments and restrictions;
- a *query generator*, for selecting which terms from the question must be necessarily present in target text passages, and which are optional;
- a *snippet searcher*, for searching corpora, using the queries, for retrieving text snippets where the candidate passage may lie;

[8] Esfinge's web site can be found at `http://www.linguateca.pt/Esfinge/` and Linguateca's web site at `http://www.linguateca.pt/` [Accessed: February 2017].

- an *answer extractor*, for identifying candidate answers in the text snippets, using answer patterns or type checking (against the question);
- *answer fusion*, for grouping lexically different but possible semantically equivalent answers into a single set;
- and an *answer selector*, for selecting one of the candidate answers produced by the previous module and choosing the supporting text/answer justification among previously extracted text snippets.

In the *query generator* module, RAPOSA makes use of query expansion, more precisely verb expansion, to cover a higher number of candidate answers, through the use of verb synonyms.

RAPOSA also deals with two main categories of questions: definition questions (that include questions about people, acronyms and miscellaneous objects) and factoid questions (including the subcategories of person, geopolitical entity, organization, entity, location, infra-structure, date/time, dimension, money, number and percentage). RAPOSA is affiliated with the Faculty of Engineering of the University of Porto.[9]

A visual representation of the structure of RAPOSA can be seen in Fig. 3.7, reproduced from Sarmento and Oliveira (2007).

3.2.4 IdSay

IdSay: Question Answering for Portuguese (Carvalho et al., 2009, 2012) is an open domain question answering system that uses mainly techniques from the area of IR, where the only external information that it uses, besides the text collections, is lexical information for the Portuguese language. IdSay uses a conservative approach to QA. Its main modules are:

- *question analysis*, where each question is processed to determine the question type and other information to be used later in the answer extraction module, and also where the search string for the document retrieval module is defined;
- *SWAN (Set Wikipedia ANswer)*, when a question is related to an entity found in Wikipedia, this module produces an answer based on that Wikipedia article, that is then added to the answers to be treated in the answer validation module;
- *document retrieval*, where a string based on words and entity information derived from the question is used to query the documents and produce a list of those that contain all the search string contents;

[9] The institution's web site can be found at `http://www.fe.up.pt/` [Accessed: February 2017].

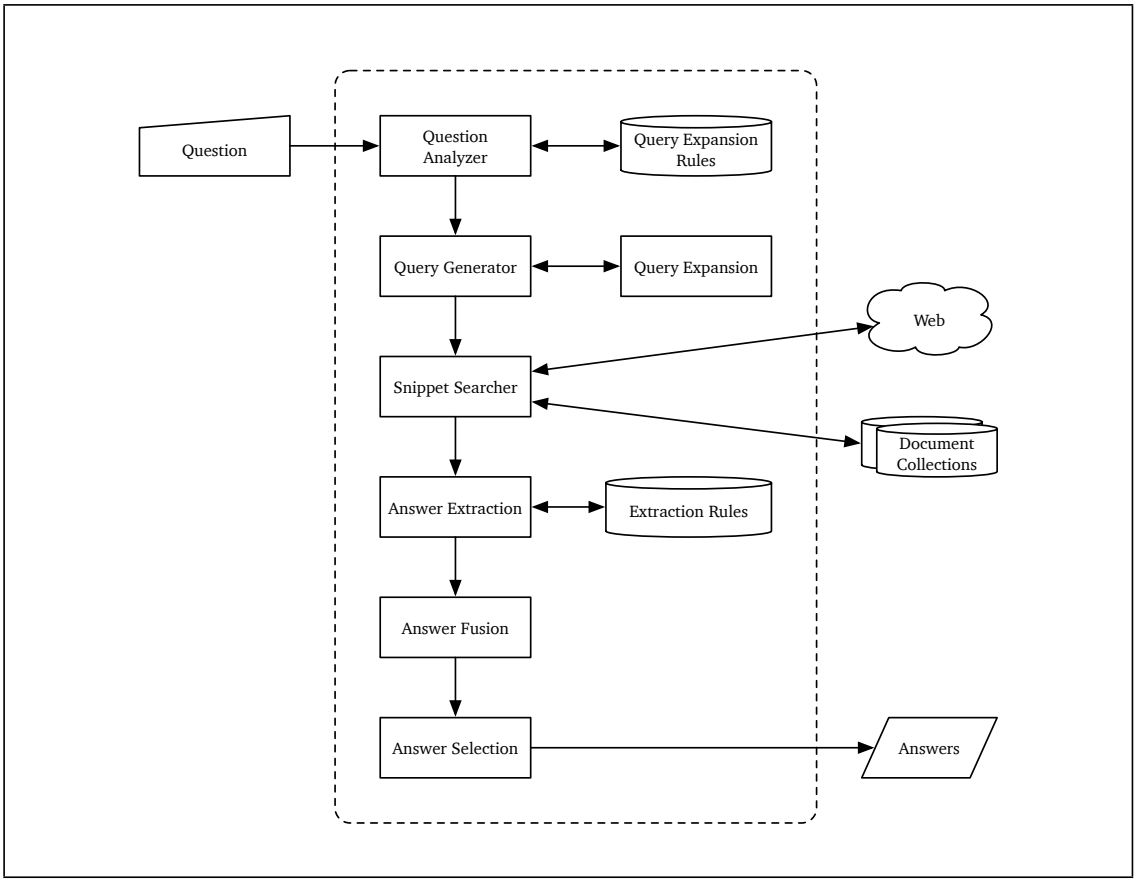

Figure 3.7: Structure and modules of RAPOSA (Sarmento and Oliveira, 2007)

- *passage retrieval*, responsible for searching passages in the documents retrieved in the previous module, up to a predefined threshold;
- *answer extraction*, where the passages are further processed, yielding short segments of text (the candidate answers);
- *answer validation*, which validates each of the candidate answers, returning the most relevant.

Whenever any of the modules is unable to produce results, the search string resulting from the *question analysis module* is revised and the process repeated.

IdSay essentially follows a classic QA system architecture. However, contrary to most QA systems, it does not store passages in the IR module, but full documents, with the passages being extracted instead at run-time. According to the authors, that solution is more flexible and suited for the case of text structures that are not clearly defined, as happens in speech transcripts, allowing iterative modification of search strings when the results are unsatisfactory. IdSay is affiliated with Universidade Aberta (The Open University of Portugal).[10]

[10]The institution's web site can be found at `http://www.univ-ab.pt/` [Accessed: February 2017].

The IdSay system architecture can be observed in Fig. 3.8, reproduced from Carvalho et al. (2012), detailing its modules and workflow.

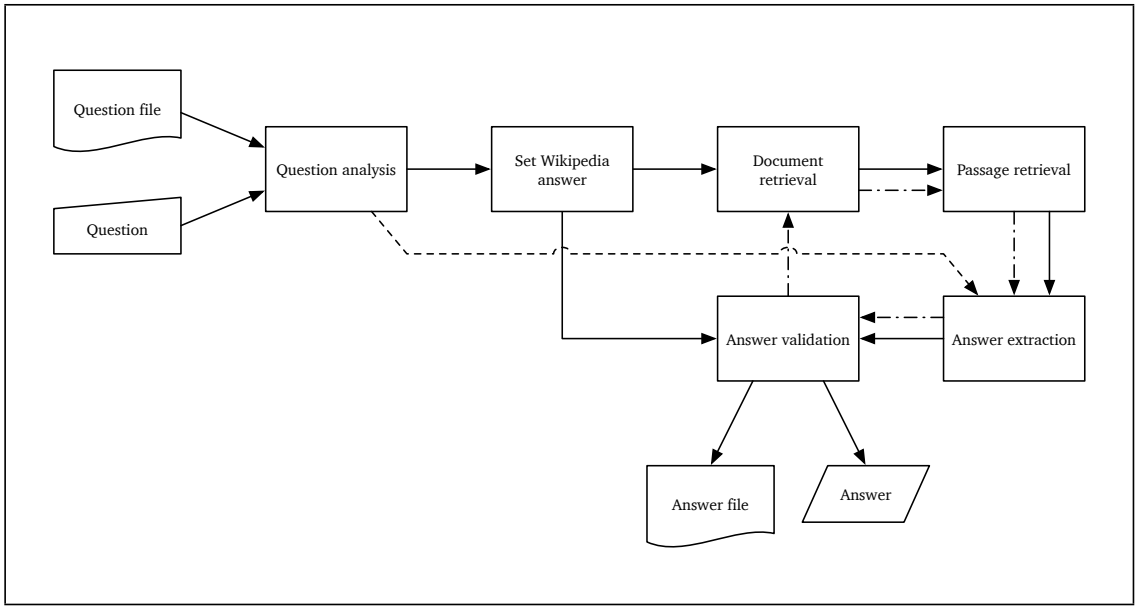

Figure 3.8: IdSay's system architecture (Carvalho et al., 2012)

3.2.5 QA@L^{2}F

QA@L^{2}F (Coheur et al., 2008; Mendes et al., 2007, 2008), the question answering system from L^{2}F, INESC-ID,[11] is a system that relies on three main tasks: *information extraction* (also called *corpus pre-processing* by the authors in Coheur et al. (2008)), *question interpretation* and *answer finding*.

In the information extraction task, text sources are processed and analyzed in order to extract potentially relevant information (such as named entities or relations between concepts), which is stored in a knowledge database. This task includes tokenization, morphological analysis and disambiguation, chunking, dependency relations extraction, and NER. Then, in the question interpretation task, the questions are processed and analyzed, selecting which terms should be used to build a SQL query to search the database. The retrieved records are then processed in the answer finding task, which selects the answer according to the question type alongside other strategies.

The QA@L^{2}F's architecture is depicted in Fig. 3.9, reproduced from Mendes et al. (2007), detailing the modules that correspond to the three main stages of that system.

[11] The institution's web site can be found at `http://www.inesc-id.pt/` [Accessed: February 2017].

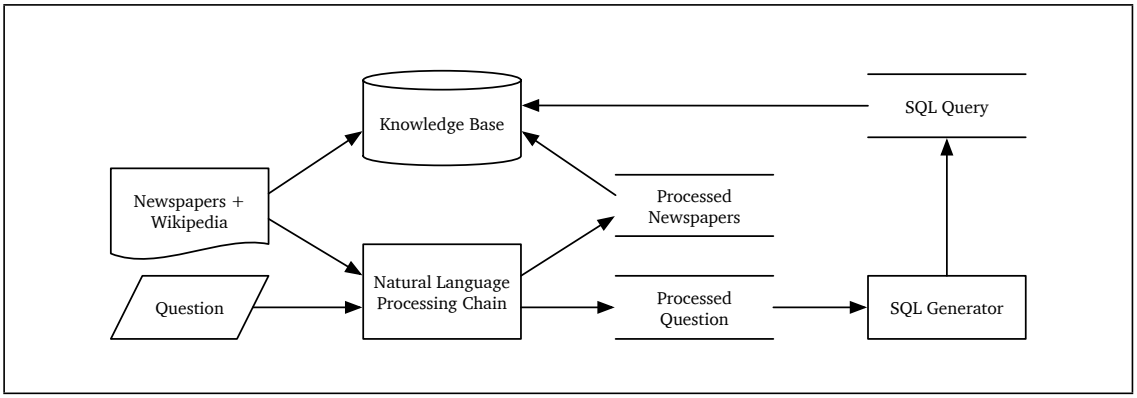

Figure 3.9: QA@L^{2}F's general architecture (Mendes et al., 2007)

3.2.6 Priberam's QA System

Some of the most well known works on NLP and QA for Portuguese have been done at Priberam.[12] Priberam's QA System for Portuguese (Amaral et al., 2005) uses a somewhat conservative approach, divided in five major modules that are described next:

- *indexing*: each document is processed through a sentence analyzer, and each sentence is classified with one or more categories through the use of answer patterns; the results are stored in multiple indices, for lemmas, heads of derivation, named entities and fixed expressions, question categories and ontology domains (regarding the documents);

- *question analysis*: questions also undergo sentence analysis, being characterized through the use of question patterns, and processed for extracting pivots (words, named entities, phrases, dates and abbreviations); question expansion is performed using heads of derivation and synonyms;

- *document retrieval*: in each document, the words are weighted according to their part-of-speech, lexical frequency, and document frequency, which add up to a document score, used to select the documents scored above a predefined threshold;

- *sentence retrieval*: sentences in the retrieved documents undergo sentence analysis, and are scored according to matches against pivots, partial matches, order and proximity of pivots, common categories between question and sentence, score of document containing the sentences, once again being selected those above a predefined threshold;

[12] Priberam Informática, S.A., is a Portuguese software company specialized in the development of tools for the areas of linguistics, juridical databases and health technologies. Its web site can be found at `http://www.priberam.pt/` [Accessed: February 2017].

- *answer extraction*: candidate answers are initially scored through the use of question answering patterns, extracted, scored again using their coherence with other candidates (how similar they are in terms of contents), and then the candidate with the highest score is selected as the final answer.

Sentence analysis, used in some of the approach's modules, consists of tokenization, lemmatization, named entity recognition, ontological domain classification and syntactical parsing.

The Priberam's QA system architecture architecture is depicted in Fig. 3.10, based on Amaral et al. (2005).

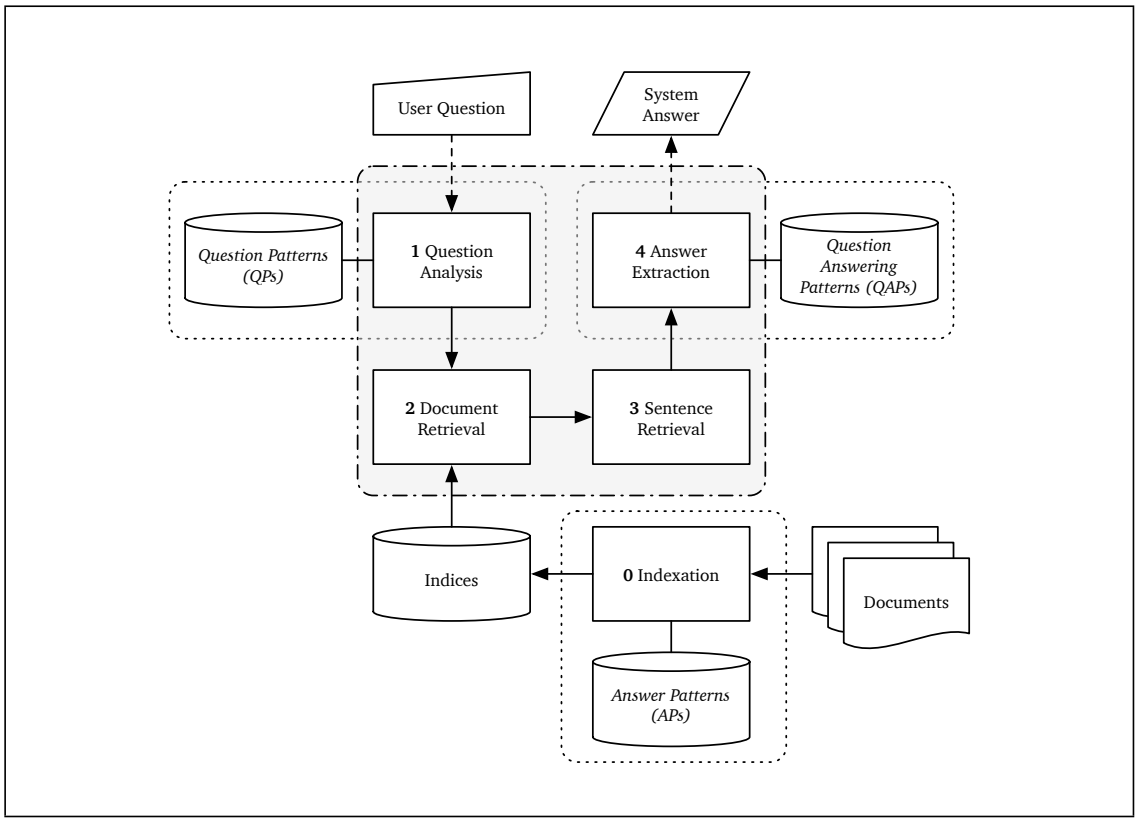

Figure 3.10: Priberam's system architecture (Amaral et al., 2005)

3.2.7 GistSumm

Brazil's Núcleo Interinstitucional de Lingüística Computacional (NILC) had built previously a summarization system, dubbed GistSumm (Pardo, 2002; Pardo et al., 2003; Balage Filho et al., 2007), that was adapted to be used in the task of monolingual QA for Portuguese texts. The text summarizer comprises three main processes: text segmentation, sentence ranking, and extract production (Balage Filho et al., 2006).

In the *text segmentation* process, the system essentially performs sentence splitting on each of the documents, with the resulting sentences being ranked in the *sentence*

ranking process, summing the frequency of its words in the whole document text. The *extract production* process deals with selecting other sentences from the document to be included in the summary, based on the co-occurrence of words between the other sentences and that with the highest rank in the previous step, guaranteeing at the same time that the candidate sentences to be added to the summary are scored above the average of all the candidates.

For the participation in QA@CLEF, some modifications were performed on GistSumm, in order to create a relation between the questions and the sentences in the summaries. Alas, two different approaches were used. One consisted of using the highest scored sentences in the summarization process and selecting the one with the best correlation against the question (through cosine measure). The other used a set of the highest scored sentences in the summarization process, and submitting them to a filter to select which should be used as the answer. That filter included POS tagging the sentences and matching the sentences against expected beginnings and other patterns (depending on the question types).

GistSumm's architecture is depicted in Fig. 3.11, based on Pardo et al. (2003).

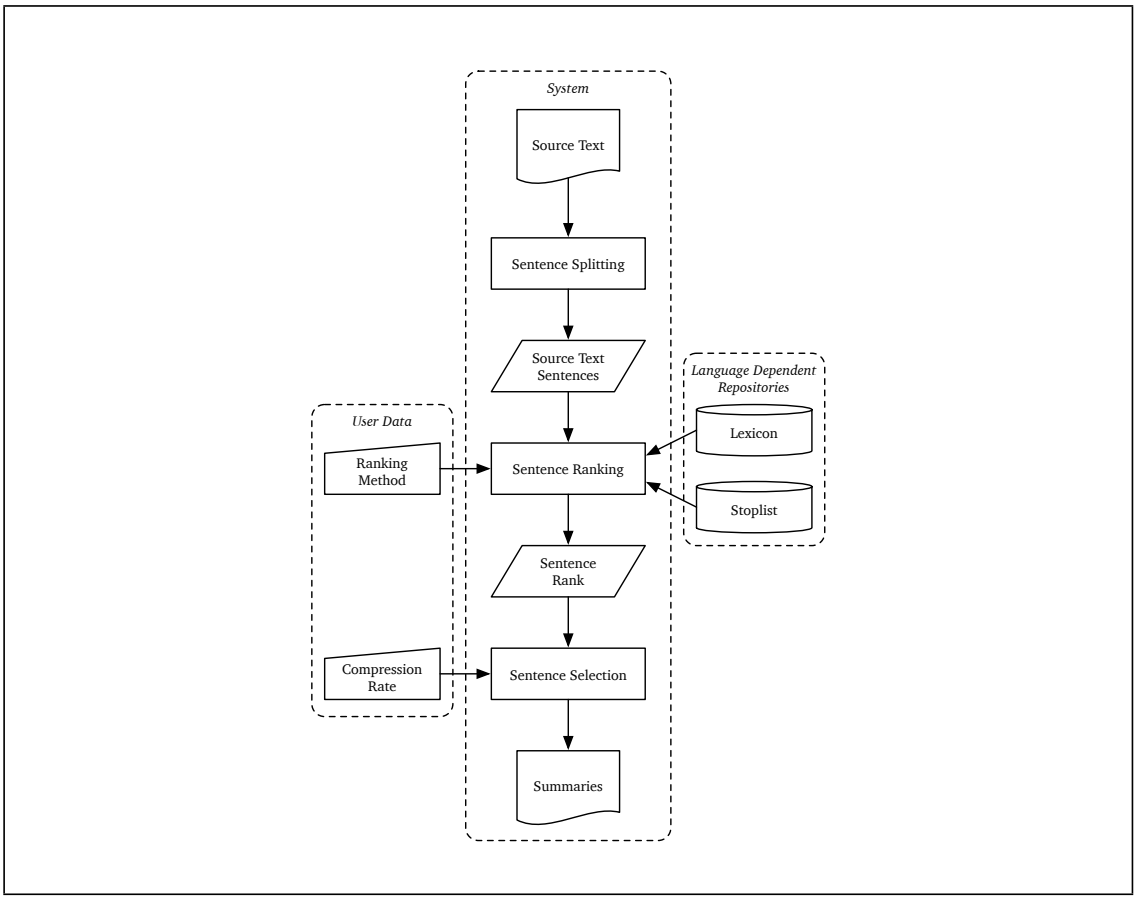

Figure 3.11: GistSumm's architecture (Pardo et al., 2003)

3.2.8 Comparison Amongst the Approaches

The easiest and more exact way of comparing the approaches would be somewhat obviously the QA@CLEF tracks, from 2004 to 2008, that used Portuguese as target language. All the described approaches have submitted data to those tracks.

As for the performance of the approaches, we present in Tables 3.3 to 3.15 the overall accuracy and other values related to results of the participants in each of the QA@CLEF tracks for Portuguese. These tables are reproduced from Magnini et al. (2005), Vallin et al. (2006), Magnini et al. (2007), Giampiccolo et al. (2008), and Forner et al. (2009), referring respectively to the years from 2004 to 2008.[13]

Some of those tables (Tables 3.3, 3.5, 3.7, 3.8, 3.10, 3.11, 3.13, and 3.14) show the number of right (R), wrong (W), incomplete inexact ($X-$), inexact with extra text ($X+$), and unsupported (U) answers, as well as the percentage of correctly answered questions (overall and separated by question types). In those same tables, the nil accuracy refers to answers that should be nil — that is, when the expected answer as no answer at all. Other tables (Tables 3.4, 3.6, 3.9, 3.12, and 3.15) break apart the results by question types. For a better a understanding of the tables and which systems are being analyzed in each of them, please refer to Table 3.2, which shows the run names associated to each system.

Table 3.2: QA systems and run names

QA Systems	Run Names
Senso	ptue0##ptpt, diue0##ptpt
Esfinge	sfnx0##ptpt, esfg0##ptpt, esfi0##ptpt
RAPOSA	uporto0##ptpt, feup0##ptpt
IdSay	idsa0##ptpt
QA@L^{2}F	ines0##ptpt
Priberam's	prib0##ptpt
NILC	nilc0##ptpt

In each year, there were several competitors: some of them have been present in almost all of the editions while others have just been present in one or two editions. Regarding the competitors, all of them had some kind of direct connection to Academia, except for Priberam's.

Starting in 2004 (Table 3.3, reproduced from Magnini et al. (2005)), the first year where Portuguese was one of the languages used on the QA@CLEF tracks, we can see the results were somewhat low, although it should be taken into account that it was a first for Portuguese systems and also for the Portuguese language, only with two participants. Additionally, only the Portuguese part of CHAVE was used (all the editions of

[13] In the transcription of the tables, in some cases there were some incongruences between tables regarding data for the same years. When it was possible, those incongruences were solved; when not, they were left as found in their sources.

Público newspaper from 1994 and 1995). Nevertheless, the best system was Senso with an accuracy close to 30%, with Esfinge lagging behind. It is worth noticing that in this first year just 199 questions were considered — there was an error creating the list of questions, and one of them had been duplicated. For comparison purposes against other languages, in 2004, the average accuracy over all the runs submitted in the monolingual tasks, that is, for each of the languages in the evaluation (Dutch, English, French, German, Italian, Portuguese and Spanish), was 23.7%, and the best result was 45.5% (Magnini et al., 2005).

Table 3.3: Comparison of the results at QA@CLEF 2004 (Magnini et al., 2005)

Run Name	R (#)	W (#)	X (#)	U (#)	Overall Accuracy (%)	Accuracy Over F (%)	Accuracy Over D (%)	NIL Accuracy P	NIL Accuracy R
ptue041ptpt	57	125	18	0	28.64	29.17	25.81	0.14	0.90
sfnx042ptpt	30	155	10	5	15.08	16.07	9.68	0.16	0.55
sfnx041ptpt	22	166	8	4	11.06	11.90	6.45	0.13	0.75

Breaking down the results according to the types of questions, we get Table 3.4 (also reproduced from Magnini et al. (2005)), being possible to notice, among other things, that the type that gets more right answers is the *factoid* regarding locations.

Table 3.4: Comparison of the results at QA@CLEF 2004 by question types (Magnini et al., 2005)

Run Name	Given Correct Answers: Definition (#) Org (14)	Per (17)	Factoid (#) Loc (43)	Man (4)	Mea (23)	Obj (6)	Org (12)	Oth (21)	Per (44)	Tim (15)	Total # (199)	%
ptue041ptpt	3	5	19	1	5	1	4	3	14	2	57	28.64
sfnx041ptpt	0	2	4	0	3	1	2	3	7	0	22	11.06
sfnx042ptpt	1	2	8	0	4	2	2	4	7	0	30	15.08
Combination	3	6	25	1	5	3	4	6	19	2	74	37.18

In 2005 (Table 3.5, reproduced from Vallin et al. (2006)), in addition to Senso and Esfinge, there was another competitor system, Priberam's, which became the best approach to Portuguese in its first participation. The other two systems had similar results to their first participation. Starting in 2005, CHAVE was fully used, contributing with both *Público* and *Folha de São Paulo* for the evaluations.

Table 3.5: Comparison of the results at QA@CLEF 2005 (Vallin et al., 2006)

Run Name	Right #	Right %	W #	X #	U #	%F (135)	Right %D (42)	%T (23)	NIL P	NIL R	NIL F
prib051ptpt	129	64.50	55	13	3	67.41	64.29	47.83	0.50	0.11	0.18
ptue051ptpt	50	25.00	125	22	3	21.48	35.71	26.09	0.10	0.67	0.18
esfg051ptpt	46	23.00	139	11	4	23.70	16.67	30.43	0.21	0.78	0.33
esfg052ptpt	43	21.50	145	10	2	23.70	14.29	21.74	0.22	0.78	0.34

Once again, for comparison purposes, in 2005, the average accuracy over all the runs submitted in the monolingual tasks was 29.36%, and the best result was 64.5%, coincidently with Priberam's for Portuguese (Vallin et al., 2006).

Breaking down the results according to the types of questions also for 2005, we get the data in Table 3.6 (reproduced also from Vallin et al. (2006)).

Table 3.6: Comparison of the results at QA@CLEF 2005 by question types Vallin et al. (2006)

Run Name	Given Correct Answers														
	Definition		Factoid						Temporarily Restricted Factoid					Total	
	Org (15)	Per (27)	Loc (30)	Mea (17)	Org (21)	Oth (15)	Per (35)	Tim (15)	Loc (5)	Mea (1)	Org (2)	Oth (6)	Per (9)	#	%
prib051ptpt	12	15	26	11	7	8	15	14	2	—	—	4	5	129	64.50
ptue05ptpt	5	10	10	1	3	3	10	2	1	—	—	2	3	50	25.00
esfg051ptpt	1	6	9	4	5	2	9	3	0	—	1	3	2	46	23.00
esfg052ptpt	0	6	8	3	3	—	13	5	0	—	1	2	2	43	21.50
Combination	12	22	28	13	11	10	27	15	3	—	1	6	7	155	77.50

In 2006 (Table 3.7, reproduced from Magnini et al. (2007)), Senso had not participated, but there were two other new participants, namely NILC and RAPOSA. However, NILC's results were scarce, and RAPOSA had some modest, but low results. Priberam's and Esfinge had results in line with the previous year, with Priberam's keeping the top position. This year was also the first with list questions, that is, questions that could accept a list of distinct answers. For example, one such question was "*Diga um jornal norueguês*,"[14] that had at least three known answers: *Aftenposten*, *Verdens Gang*, and *Tagbladet*. The return of one correct member of the list meant the answer should to be considered correct.

Table 3.7: Comparison of the results at QA@CLEF 2006 (Magnini et al., 2007)

Run Name	R (#)	W (#)	X+ (#)	X− (#)	U (#)	Overall Accy. (%)	Accy. o. F (%)	Accy. o. D (%)	NIL Accy. P	NIL Accy. R
prib061ptpt	134	58	6	1	1	67.0	65.36	72.34	43.33	72.22
esfg061ptpt	49	139	7	2	3	24.5	22.88	29.79	15.53	88.89
esfg062ptpt	45	142	6	6	1	22.5	20.26	29.79	14.95	88.89
uporto062ptpt	26	169	2	3	0	13.0	11.76	17.02	7.64	66.67
uporto061ptpt	23	177	0	0	0	11.5	9.80	17.02	8.29	77.78
nilc062ptpt	3	190	0	5	2	1.5	1.96	0.00	8.57	16.67
nilc061ptpt	0	189	1	8	2	0.0	0.00	0.00	—	—

This year, the average accuracy over all the runs submitted in the monolingual tasks was 27.94%, and the best result was 68.95% (Magnini et al., 2007).

In 2006, there was also an evaluation that accepted all submitted answers by each of the participants, up to a limit of ten answers per question, whose results are presented in Table 3.8 (Magnini et al., 2007).

Once more, the results have been broken according to the types of questions for 2006, as shown in Table 3.9 (Magnini et al., 2007)).

[14] In English, "State a Norwegian newspaper."

Table 3.8: Comparison of the results at QA@CLEF 2006 with up to ten answers (Magnini et al., 2007)

Run Name	R (#)	W (#)	X+ (#)	X− (#)	U (#)	Overall Accuracy (%)
prib061ptpt	134	58	6	1	1	67.00
esfg061ptpt	49	143	11	2	3	23.56
esfg062ptpt	45	146	7	6	1	21.95
uporto062ptpt	42	172	3	6	0	18.83
uporto061ptpt	36	178	0	0	0	16.82
nilc062ptpt	3	190	0	5	2	1.50
nilc061ptpt	0	189	1	8	2	0.0

Table 3.9: Comparison of the results at QA@CLEF 2006 by question types (Magnini et al., 2007)

Run Name	Correct Answers										Total	
	Definition (47)				Factoid (Time Restricted Question; List) (153 (27; 9))							
	Obj	Org	Oth	Per	Loc	Mea	Org	Oth	Per	Tim	#	%
	7	8	24	9	25 (2; 0)	21 (2; 0)	223 (6; 3)	30 (6; 2)	34 (11; 3)	19 (0; 1)	200 (27; 9)	
esfg061ptpt	3	3	5	3	10	4 (1; 0)	2 (1; 0)	6 (0; 0)	11 (2; 1)	3 (0; 0)	49 (4; 1)	24.5
esfg062ptpt	3	4	5	2	9	4 (1; 0)	1 (1; 0)	6 (1; 0)	8 (2; 1)	3 (0; 0)	45 (4; 1)	25.5
nilc061ptpt	0	0	0	0	0	0	0	0	0	0	0	0.0
nilc062ptpt	0	0	0	0	1	1 (1; 0)	0	0 (1; 0)	0	1 (0; 0)	3 (1; 0)	1.5
prib061ptpt	5	6	14	8	17 (1; 0)	13 (1; 0)	17 (3; 3)	17 (1; 1)	20 (4; 3)	16 (0; 1)	134 (10; 8)	67.0
uporto061ptpt	1	0	2	5	3	3 (1; 0)	1 (1; 0)	2 (0; 0)	5 (2; 1)	1 (0; 0)	23 (4; 1)	11.5
uporto062ptpt	1	0	2	5	4	3 (1; 0)	1 (1; 0)	3 (0; 0)	4 (2; 1)	3 (0; 0)	26 (4; 1)	13.0
Combination	5	7	19	9	19 (1; 0)	14 (1; 0)	17 (3; 3)	19 (1; 1)	23 (5; 3)	17 (0; 1)	149 (11; 8)	74.5

In the following year, 2007 (Table 3.10, reproduced from Giampiccolo et al. (2008)), the participants were once more Priberam's, Esfinge, Senso, and RAPOSA, with the addition of QA@L^{2}F. The results were similar to previous years, with QA@L^{2}F ending in front of Esfinge. Senso and Priberam's kept their relative positions, although Senso has improved its results, and Priberam's has lowered its to just above 50%.

Once again, for comparison purposes, in 2007, the average accuracy over the all runs submitted in the monolingual tasks was around 23%, and the best result was around 53% (Giampiccolo et al., 2008).[15]

This year was also the first year that Wikipedia was used alongside CHAVE. Of the 200 questions, during the selection phase, 132 had their answers in Wikipedia, 55 in CHAVE, and 13 were nil (had no answer in the corpora). However, after the evaluations, of the 200 questions, 159 had answers found in Wikipedia, 62 in CHAVE, and 13 were still nil (Giampiccolo et al., 2008). That is, there were questions that ended up having answers in both corpora.

Beyond the novelty of introducing Wikipedia as a corpus, there was another significant change: 51 of the 200 questions were linked questions. That is, for determining the *full* question, the system had to analyze either the previous question or its answer. For example, there was a question whose text "*Qual é a mais pequena delas?*" could only be understood knowing the text of the previous question "*Quais são as sete colinas de*

[15] Unfortunalely, the chart in the paper cited just lets us do an approximation of the two values.

Table 3.10: Comparison of the results at QA@CLEF 2007, all questions (Giampiccolo et al., 2008)

Run Name	R (#)	W (#)	X+ (#)	X– (#)	U (#)	Overall Accuracy (%)	NIL Accuracy P (%)	NIL Accuracy R (%)
prib071ptpt	101	88	5	5	1	50.5	27.8	46.2
diue071ptpt	84	103	1	11	1	42.0	11.7	92.3
feup071ptpt	40	158	1	1	0	20.0	8.3	84.6
ines072ptpt	26	168	0	4	2	13.0	7.2	84.6
ines071ptpt	22	171	1	4	2	11.0	7.3	69.2
esfi071ptpt	16	178	0	4	2	8.0	6.3	69.2
esfi072ptpt	12	184	0	2	2	6.0	6.1	84.6

Roma?"[16] The distinction between the results of both types of questions is presented in Table 3.11, also reproduced from Giampiccolo et al. (2008).

Table 3.11: Comparison of the results at QA@CLEF 2007, separated questions (Giampiccolo et al., 2008)

Run Name	First (Unlinked) Questions (149) R #	W #	X+ #	X- #	U #	Accy. %	Linked (51) R #	Accy. %
diue071ptpt	61	77	1	9	1	40.9	23	45.1
esfi071ptpt	11	132	0	4	2	7.4	5	9.8
esfi072ptpt	6	141	0	1	1	4.0	6	11.8
feup071ptpt	34	113	1	1	0	22.8	6	11.8
ines071ptpt	17	125	1	4	2	11.4	5	9.8
ines072ptpt	21	122	0	4	2	14.1	7	13.7
prib071ptpt	92	86	3	5	1	61.7	9	17.6

The results separating the questions according to their types can be found in Table 3.12 (adapted from Giampiccolo et al. (2008)).

Table 3.12: Comparison of the results at QA@CLEF 2007 by question types (Giampiccolo et al., 2008)

Run	Correct Answers: Definitions Obj 6	Org 6	Oth 9	Per 9	Cou 21	Loc 34	Mea 16	Factoids (Including Lists) Obj 5	Org 22	Oth 28	Per 24	Tim 20	Total # 200	%
diue071ptpt	6	4	5	4	11	17	4	3	6	8	7	9	84	42.0
esfi071ptpt	1	0	0	0	3	3	0	0	1	0	1	7	16	8.0
esfi072ptpt	1	0	0	0	2	4	0	0	1	0	2	2	12	6.0
feup071ptpt	3	2	4	7	4	8	0	0	3	1	3	5	40	20.0
ines071ptpt	4	4	6	0	1	3	0	0	0	0	2	2	22	11.0
ines072ptpt	5	5	6	0	2	4	0	0	0	0	2	2	26	13.0
prib071ptpt	6	4	6	7	9	15	10	1	11	14	8	10	101	50.5
Combination	6	5	8	9	16	24	12	3	12	17	12	13	137	68.5

The year of 2008 (Table 3.13, reproduced from Forner et al. (2009)) was the last one with QA tracks with this specific characteristics. Like in the previous year, there were

[16] Translated loosely into English, the questions read as "Which of them is the smallest?" and "Which are the seven hills of Rome?"

also 51 linked questions, and it had 12 questions with nil answers. The participants were the same of the previous year, with the addition of IdSay. The results were also similar, with Priberam's and Senso improving their results over the previous edition, and IdSay introducing itself in the third position.

Table 3.13: Comparison of the results at QA@CLEF 2008, all questions (Forner et al., 2009)

Run Name	R	W	X+	X–	U	Overall Accuracy	NIL Accuracy		
	(#)	(#)	(#)	(#)	(#)	(%)	#	P (%)	R (%)
prib081	127	55	9	3	4	63.5	8	12.5	10
diue081	93	94	8	1	2	46.5	21	9.5	20
idsa081	65	119	8	0	8	32.5	12	16.7	20
esfi081	47	134	5	7	5	23.5	20	20.0	20
ines081	40	150	2	1	5	20.0	123	9.7	90
ines082	40	150	2	1	5	20.0	123	9.7	90
esfi082	39	137	7	9	6	19.5	20	15.0	10
feup081	29	165	2	2	2	14.5	142	8.5	90
feup082	25	169	3	1	2	12.5	149	8.1	90

In 2008, the average accuracy over the all runs submitted in the monolingual tasks was 23.63%, and the best result was around 63.5% (Forner et al., 2009).

The distinction between the results of both unlinked and linked questions is presented in Table 3.14, also reproduced from Forner et al. (2009).

Table 3.14: Comparison of the results at QA@CLEF 2008, separated questions (Forner et al., 2009)

Run	First Questions (149)						Linked (51)	
	R	W	X+	X-	U	Accy.	R	Accy.
	#	#	#	#	#	%	#	%
diue081ptpt	82	59	6	3	1	54.3	11	22.4
esfi081ptpt	42	92	5	7	5	27.3	7	14.3
esfi082ptpt	33	97	6	9	6	21.9	8	16.3
feup081ptpt	29	116	2	2	2	19.2	3	6.1
feup082ptpt	25	120	3	1	2	16.6	3	6.1
idsa081ptpt	54	85	6	0	6	35.8	11	22.4
ines081ptpt	35	106	2	3	5	23.2	8	16.3
ines082ptpt	35	106	2	3	5	23.2	8	16.3
prib081ptpt	105	32	9	4	1	69.5	11	44.9

Finally, the results separating the questions according to their types can be found in Table 3.15 (adapted from Forner et al. (2009)).

As we can observe overall from 2004 to 2008, the best system is Priberam's, with Senso as runner up, and the other systems much behind. It is also evident that only Priberam's was able to go over the 50% barrier on most of the editions, with Senso close, but never over that barrier.

Table 3.15: Comparison of the results at QA@CLEF 2008 by question types (Forner et al., 2009)

Run	Definitions						Factoids (Including Lists)							Total	
	Loc	Obj	Org	Oth	Per	Cou	Loc	Mea	Obj	Org	Oth	Per	Tim	#	%
	1	6	6	8	6	17	38	16	2	10	33	33	24	200	
diue081ptpt	0	5	6	8	5	6	17	8	1	5	13	8	11	93	46.5
esfi081ptpt	0	1	2	4	2	8	8	2	0	2	2	14	4	49	24.5
esfi082ptpt	0	0	0	1	1	8	8	2	0	2	2	13	4	41	20.5
feup081ptpt	0	1	1	1	1	5	4	4	0	1	2	8	4	32	16
feup082ptpt	0	1	1	1	1	5	3	4	0	1	2	6	3	28	14
idsa081ptpt	1	5	1	5	5	9	9	9	0	0	6	8	7	65	32.5
ines081ptpt	1	5	1	7	3	4	9	2	0	0	1	4	6	43	21.5
ines082ptpt	1	5	1	7	3	4	9	2	0	0	1	4	6	43	21.5
prib081ptpt	0	5	5	6	2	11	21	13	1	7	18	22	16	127	63.5
Combination	1	6	6	8	6	16	31	15	1	7	23	27	21	168	84

3.3 Other Approaches

The next approaches are described under the classification "other approaches" as there is no way to assess their performance (unlike what happened with those described previously, that have been submitted to the QA@CLEF's Portuguese tracks). Nevertheless, we felt they should be mentioned, as they are clearly in the scope of the task addressed in this thesis.

3.3.1 XisQuê

XisQuê (Branco et al., 2008a,b) is an online QA service for Portuguese. It is a real-time, open-domain question answering system for documents written in Portuguese and accessible through the World Wide Web. The system resorts to a number of shallow processing tools and question answering techniques that are specifically deasigned to cope with the Portuguese language. It is composed of four main stages:

- **question processing**, which includes the extraction of the main verb, using it as the pivotal element for identifying the major supporting noun phrase (NP) of the question; the detection of the expected semantic type of the answer, based on the named entities found in the question; and the extraction of relevant keywords, such as verbs, nouns and entities;

- **document retrieval**, using search engines such as *Ask*, *Google*, *MSN Live*, and *Yahoo!*, whose results would be then processed, downloading the linked documents and processing their contents;

- **answer extraction**, including candidate selection, ranking the candidate answers based on the keywords present in them and using an heuristic for defining the threshold to separate the "best" candidates from the remaining, and answer ex-

traction properly, based on the named entities and their types found in the candidate answers, and in extraction patterns, using phrases and verbs;

- and also a **web service** that acts as a front-end for system, available online at `http://xisque.di.fc.ul.pt/` (as of February 2017).

3.3.2 A QA System for Portuguese

This QA system for Portuguese (Prolo et al., 2005) is an approach whose core is the unification of open predicates expressed by the questions and closed predicates conveyed by the sentences in the corpus, using Prolog-like predicates — alas, the approach uses Prolog as its base. For instance, the question "Who killed Odete Roitman?,"[17] retrieved from Prolo et al. (2005), translates into **`kill`**`(`*`Who`*`, Odete Roitman)`, and the possible answer "Laura killed Odete Roitman" translates into **`kill`**`(Laura, Odete Roitman)`. The system then tries to match predicates. The system's main steps follow: documents are syntactically analyzed by a parser; semantic information is extracted from the syntactical structure; semantic/pragmatic interpretation is obtained from the semantic representation, and from an ontology of concepts; the resulting information is used to help an information retrieval system to select sets of relevant documents in the query-answering process.

3.3.3 A QA System for Portuguese Juridical Documents

The Question Answering system for Portuguese juridical documents has a very specific scope, being "applied to the complete set of decisions from several Portuguese juridical institutions (Supreme Court, High Court, Courts, and Attorney-General's Office) in a total of 180,000 documents" (Quaresma and Rodrigues, 2005b).

This system aims at answering four types of questions: places, dates, definitions and specific. For that, the system relies, in a first phase, on processing the existing documents, by IR indexing, applying a Portuguese parser, making semantic interpretation, creating an ontology and by semantic/pragmatic interpretation. In a second phase, the system executes the question answering in three steps: first, each query is analyzed; second, a potentially relevant set of documents is selected by means of a retrieval system; and third, for each selected document, the semantic pragmatic/representation of the query is evaluated in the document semantic/pragmatic representation.

[17] Out of curiosity, this question comes from the plot of *Vale Tudo*, one of the most famous Brazilian soap operas of all time, which, in 1989, made headlines all over the country.

3.4 PÁGICO

Quoting the organization, "PÁGICO [was] a joint assessment in the area of information retrieval for Portuguese, that [intended] to evaluate systems that find non trivial answers to complex information needs in Portuguese" (Santos, 2012). PÁGICO was eventually a task of information retrieval over a snapshot of the Portuguese version of Wikipedia, and it is on that context that it was developed and applied the RAPPORTÁGICO approach, an initial incarnation of RAPPORT adapted for PÁGICO. One of the most distinctive features of PÁGICO is that it was open to both human and automatic (computer systems) competitors. PÁGICO was organized by Linguateca with the collaboration of the University of Oslo, the Pontifical Catholic University of Rio de Janeiro, and the University of Coimbra.

There were 150 topics that could be formulated as questions, statements or even commands. The topics were grouped in themes, and further divided in sub-themes: Literature (history, literature, linguistics, journalism, philosophy), Arts (music, movies, television, plastic arts, arts), Geography (geography, architecture/urbanism, demography, geology), Culture (anthropology/folklore, religion, culinary, culture, education), Politics, Sports and Science (health, zoology, science, botany, geology, mathematics) and Economy (Mota et al., 2012).

The corpus used was the Portuguese Wikipedia, using a dump from November 2011, whose main characteristics have already been addressed in Subsection 3.1.1.

Fig. 3.12 presents an example of a topic (question) used in PÁGICO and associated candidate answers. Notice that the answer provides an article page and why (sometimes another page) it should be considered.

```
[Pergunta]
ID: 1; Descrição:  Filmes sobre a ditadura ou sobre o golpe militar no Brasil; Grande tema:  Artes,
Letras; Tema:  cinema, história; Lugar:  br

[Respostas]
Correcta:  1; Justificada:  1; ID: Pagico_001; Grande tema:  Artes, Letras; Tema:
Cinema, História; Local:  Brasil; Resposta:  pt/l/a/m/Lamarca.259536.xml; Justificação:
pt/a/n/e/Anexo_Lista_de_filmes_sobre_ditaduras_militares_na_América_Latina.605516.xml

Correcta:  1; Justificada:  1; ID: Pagico_001; Grande tema:  Artes, Letras; Tema:  Cinema,
História; Local:  Brasil; Resposta:  pt/q/u/a/Quase_Dois_Irmãos.c80d70.xml; Justificação:
pt/a/n/e/Anexo_Lista_de_filmes_sobre_ditaduras_militares_na_América_Latina.605516.xml

Correcta:  1; Justificada:  1; ID: Pagico_001; Grande tema:  Artes, Letras; Tema:  Cinema, História;
Local:  Brasil; Resposta:  pt/o/_/a/O_Ano_em_que_Meus_Pais_Saíram_de_Férias.b58539.xml; Justificação:
pt/a/n/e/Anexo_Lista_de_filmes_sobre_ditaduras_militares_na_América_Latina.605516.xml
```

Figure 3.12: A PÁGICO topic and associated answers

At the moment of the writing of this thesis, PÁGICO had just one edition in 2012, with a summary of the results being presented in Table 3.16. This table takes data retrieved

from two tables found in Mota (2012), from the most general scenario addressing all topics, presenting the numbers of: answered topics ($|T|$), provided answers ($|R|$), the ratio of answers per topic ($|R|/|T|$), right and correctly justified answers ($|C|$), and right but incorrectly justified answers ($|\tilde{C}|$). The table also presents the values for precision (P), *pseudo-recall* (α), which, in addition to the *a priory* known answers, also considers the right answers found during the evaluation of each system, and a final score (M) resulting from the product of the number of right answers ($|C|$) and the precision value (P). The results are ordered by the score of each participant.

PÁGICO was, in many respects, different from the QA@CLEF tracks, mainly due to the type of answers — Wikipedia pages (addresses) *vs.* text passages — and for the competitors — both humans and computer systems *vs.* just computer systems. Noteworthy was also the fact that there were only two computer systems competing against humans: RAPPORTÁGICO and RENOIR.

Table 3.16: Results of PÁGICO (Mota, 2012)

Contestant	$\|T\|$	$\|R\|$	$\|R\|/\|T\|$	$\|C\|$	$\|\tilde{C}\|$	P	α	M
ludIT	150	1387	9.25	1065	34	0.768	0.474	817.754
GLNISTT	148	1016	6.86	661	52	0.651	0.294	430.04
João Miranda	40	101	2.52	80	3	0.792	0.036	63.366
Ângela Mota	50	157	3.14	88	3	0.56	0.039	49.325
RAPPORTÁGICO (3)	114	1730	15.18	208	13	0.12	0.092	25.008
RAPPORTÁGICO (2)	115	1736	15.1	203	13	0.117	0.09	23.738
RAPPORTÁGICO (1)	116	1718	14.81	181	11	0.105	0.08	19.069
Bruno Nascimento	18	34	1.89	23	1	0.676	0.01	15.559
RENOIR (1)	150	15000	100	436	38	0.029	0.194	12.673
RENOIR (3)	150	15000	100	398	29	0.026	0.177	10.56
RENOIR (2)	150	15000	100	329	25	0.022	0.146	7.216

Not surprisingly, humans had overall a better score than computer systems. This can be explained by several reasons. For example, humans, whenever they find the right answers, typically stop searching form more answers, reducing significantly the number of submitted answers. In computer systems, the number of submitted answers is usually higher than that of humans. Nevertheless, RAPPORTágio (all three runs considered) managed to be the 5th out of 7 competitors, in front of a human and of the other computer approach.

3.4.1 RAPPORTÁGICO

RAPPORTÁGICO (Rodrigues et al., 2012) was based on the syntactic analysis of texts and on the identification of synonyms of context words (through a lexical ontology). The approach was divided in four distinct stages:

1. Indexing of the Wikipedia articles contents;

2. Analysis and processing of the topic sentences (questions, for all purposes), focusing on the frases that constituted them;

3. Search on the contents index, using queries stemming from the previous stage, and identifying the corresponding articles;

4. Processing of answers.

Essentially, the system started by storing and indexing, using Lucene (McCandless et al., 2010), all data regarding the texts of the Wikipedia articles, including stemmed versions of the texts. Then, the topics were likewise processed, generating a base query and possible alternatives, that were matched against the index. Matching texts were ordered and presented as candidate answers, by means of their identifier. A detailed description of RAPPORTÁGICO is presented in the next chapter (Chapter 4).

3.4.2 RENOIR

RENOIR is an extension of a NER system called REMBRANDT (Cardoso, 2012b), its name being an acronym for "REMBRANDT's Extended NER On IR interactive retrievals." For the participation in PÁGICO, RENOIR incorporates features from a geographic information retrieval (GIR) system from the same author (Cardoso et al., 2008), although mainly as an auxiliary structure for storing and retrieving data (Cardoso, 2012a). In practice, there are two major modules in this approach: RENOIR for reformulating the queries and GIR for information storage and retrieval.

One key aspect of RENOIR is that it does not use Wikipedia directly, but DBpedia (Bizer et al., 2009), being its key modules an *interpreter* and a *reasoner*, that process the questions, looking, for instance, for entities and then building a SPARQL query to be run against DBpedia. The system starts by analyzing the topic using the interpreter that converts the questions into objects that represent the multiple properties of the topic — a theme (the question type), conditions (a list of criteria that filters the candidate answers), and the expected answer type (which sets the properties a candidate question must have). After an object is produced, it is used to build queries to run against a knowledge-base, by the reasoner. The reasoner starts by running the query that most resembles the topic, and, depending on its results, may loose some of the restrictions until it gets acceptable results (other than zero).

The knowledge-base used is where the corpus documents are stored and indexed, together with other data sources, namely DBpedia, *Yahoo! GeoPlanet*,[18] and Wikipedia. The system is depicted in Fig. 3.13, reproduced from Cardoso (2012a).

[18] *Yahoo! GeoPlanet* can be accessed at: `https://developer.yahoo.com/geo/geoplanet/` [Accessed: February 2017].

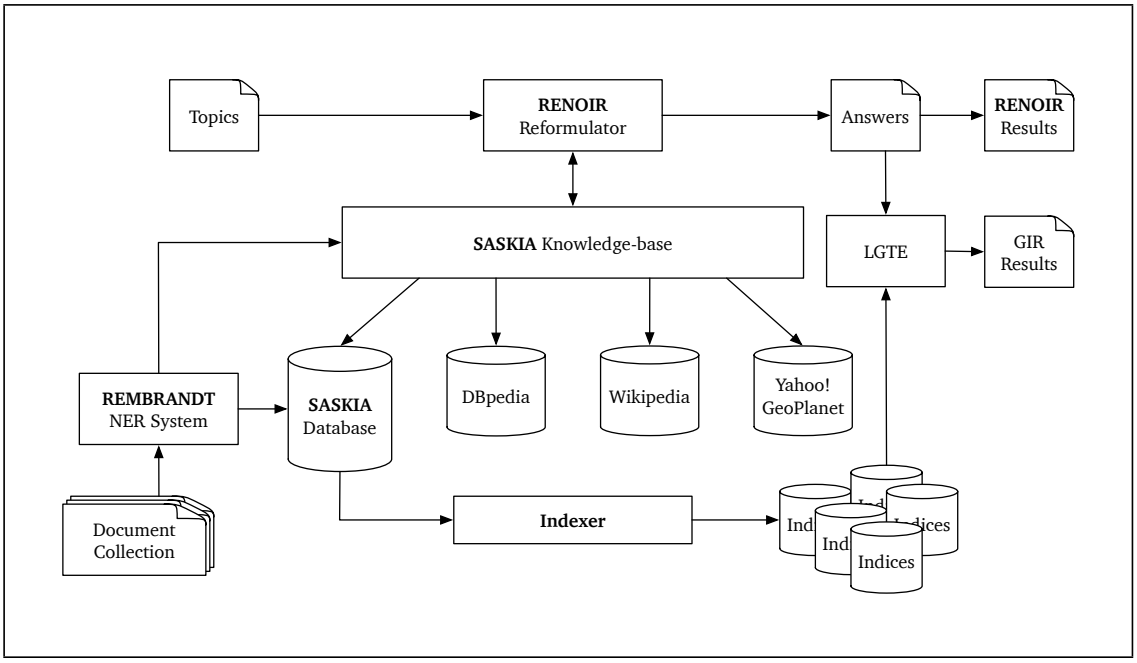

Figure 3.13: RENOIR's system used in PÁGICO (Cardoso, 2012a)

3.5 Global Analysis of Related Work

Here, we present a global analysis of the multiple approaches described in this chapter, pointing out theirs strengths and weaknesses. The main conclusions are the following:

- All QA systems use some kind of document indexing implemented in the approaches, resorting to relational databases or similar tools, or use already made indices, as is the case of web search engines;
- Some kind of text normalization always occurs, with some approaches using stemming and others opting for lemmatization;
- Query expansion is usual on most systems, some of them using local knowledge bases, which can also be used to provide quick answers;
- Query reformulation is also common, turning questions into answer beginnings, based on models;
- Data extracted or retrieved from corpora are stored in indices or using some logic module to describe the data;
- In most systems, document retrieval and passage retrieval are two distinct steps, as is answer extraction;
- Not every approach performs the classification of questions dividing them into different categories;

- The answers retrieved by a system are usually ranked, specially when there are many available, before being presented to the user.

The systems with better results usually have well defined stages involving text processing, question analysis, document retrieval and answer extraction, sometimes with subdividing some of these stages.

Apparently, the richer the text processing stage, the better the system performs. Also, there is usually a symmetry between the text processing and the question processing stages. Models of question-answer pairs are used in multiple systems, but the models alone seem to be somewhat limited, although improving, for instance, the categorization of both sentences and questions.

Finally, just storing and indexing the corpora seems to be not enough, introducing much noise in the process.

Chapter 4

RAPPORTÁGICO

In this chapter we present RAPPORTÁGICO, the system that was, retrospectively, the first real test for some of the ideas behind what would eventually become the current RAPPORT approach (which is thoroughly described in next the chapter).

The development of RAPPORT was in its early stages when we have learned of the PÁGICO competition. With our participation in that competition we were hoping to test our approach and also to get some experience that could be used later when testing RAPPORT with Wikipedia in QA@CLEF (alongside CHAVE). Although some adaptations had to be made, it was in the context of PÁGICO that some of the major ideas behind the RAPPORT approach were first tested, under the name RAPPORTÁGICO.

The decision to participate in PÁGICO came from it being the first assessment specifically targeted for information retrieval over the Portuguese Wikipedia. Although our main interests were initially different (focusing just on question answering in its stricter scope), there were aspects in common with the objectives of PÁGICO. Those objectives were found to be "between information retrieval and question answering," which could be "further described as answering questions related to Portuguese-speaking culture in the Portuguese Wikipedia, in a number of different themes, and geographic and temporal angles" (Mota et al., 2012).

We have made the choice of describing RAPPORTÁGICO and RAPPORT and in two separate chapters, despite the similarities between them, because the processing of the corpora, the questions, and respective answers end up being different on both approaches. In RAPPORTÁGICO the goal is to present the documents that contain the answers themselves — or, more precisely, the documents are the answers —, while in RAPPORT the system has to deal with the processing of the documents' contents for retrieving just the passages that constitute the intended answers. That being said, RAPPORT includes many of the tasks of RAPPORTÁGICO, including improvements, as well as additional tasks. That is to say, RAPPORT's tasks are a superset of RAPPORTÁGICO's.

4.1 Overview

RAPPORTÁGICO is an approach based on the indexing of Wikipedia articles, on identifying phrases in the sentences of the topics, and its subsequent processing and analysis, in order to facilitate the matching between topics and articles that can serve as answers. The phrases facilitate the identification of structures with different roles within the topic sentence. Before being used to query the index, phrases found in the topics undergo some kind of manipulation, such as the expansion of the words that constitute words of similar meaning (synonyms).

The RAPPORTÁGICO approach to PÁGICO can be split into four different parts that bear some similarities with the current version of RAPPORT:

1. **Indexing** of articles contents;

2. **Analysis** and **processing** of topic sentences, which can be viewed as questions, with emphasis on the phrases that comprise them;

3. **Searching** on the contents index, using queries generated in the previous step, and identification of the articles corresponding to the intended answers;

4. **Processing** of the answers.

Each of these four distinct parts is described next, followed by the major implementation details. A visual representation of RAPPORTÁGICO is also depicted in Fig. 4.1.

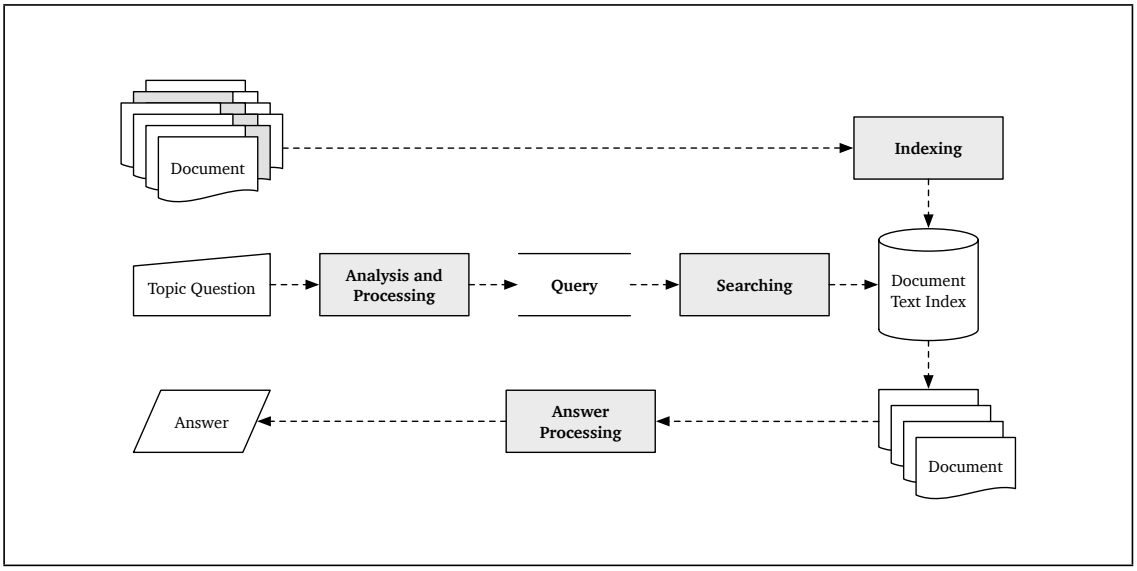

Figure 4.1: RAPPORTÁGICO's system architecture

4.2 Indexing

The first step consists of the indexing of all the articles found in the Portuguese version of Wikipedia, available in a *collection* produced for PÁGICO (Simões et al., 2012), with the contents of each article being stemmed prior to storage.

Stemming is useful because, that way, verbal inflections, as well as the gender and number of nouns and adjectives, can be ignored and stored under the same representation. For example, the inflections `vence` (*he wins*), `venceram` (*they won*), and `venceremos` (*we will win*) all end up normalized the same way as `venc` — which, being the stem of the inflections, is not a valid dictionary form.

This increases the number of matches between the queries and article contents, even if only by the fact that, at the verbal inflections level, there could be different tenses between the topics and the article. The same can be said about noun number, adjective degrees, or noun gender. That was a major reason at the time for us to discard the use of an approach without a stemmer.

Although stemming brings worthwhile advantages along with it, it also comes along with some disadvantages. Beyond an increase in ambiguity, the major handicap of stemming is that it processes all words the same way, regardless of their part-of-speech classification.

To avoid this problem, we intended to normalize words through lemmatization, using one of the initial versions of LEMPORT. However, lemmatization over the Wikipedia's collection used, at the time, was shown to be too time consuming in the indexing stage, even if performed offline, which would impair our experimentations within the approach, which led us to abandon that path, and adopting the stemming tool provided by Lucene. LEMPORT had just begun to be developed and had yet to be time optimized and improved.

Finally, for indexing, we have opted for using the Lucene search engine, which allowed for the creation of a document index with two fields: the address (path and name), and the contents of the article. Using Lucene for the index accounts for two advantages:

- It provides a way of storing the articles contents while, at the same time, it makes the document search easier and faster in terms of queries made through text, allowing for full text searches;

- It allows the words to be normalized while processing the articles' contents. In this case, we used the *Portuguese analyzer* (a stemmer, available in the Lucene contributions) for obtaining the stems of the words, which enables a wider match between the queries and the index entries.

4.3 Topic Analysis and Processing

On the second step of the approach, the topic sentences (the questions) went through multiple levels of processing, with the aim of building a query in order to search the document index. As it happened with the storage of the contents, the topics were also stemmed, for the same reasons.

In addition to stemming, the topics underwent phrase chunking, based on the conviction that it would be easier to handle words in groups, where some kind of relation holds them together, rather than to process them individually as a set of unrelated terms, other than belonging to the same sentence and following some grammatical rules. Phrase chunking used a previous chunking method implemented by us, specified by rules, and that depended also on POS tagging and NER.

4.3.1 Phrase Identification

For chunking (or phrase identification), we adopted the use of the noun phrases (NP) and verb phrases (VP) that are present in the topics. Based on the output of chunking, we have defined a heuristic for the most common cases, that helped to recognize the most important elements of the topics: the first noun phrase (and the nouns in it) are the target or topic category, while the verb phrase and other noun phrases define restrictions over the category.

To get the phrase chunks of each topic sentence, we have resorted to POS tagging each of the tokens in the sentences, and then grouping them in chunks according to their tags, performing the two following steps:

- POS tagging (together with NER), based on the POS tagger of the *OpenNLP* toolkit and on the use of models trained for Portuguese, also made available from the same project. An example of the annotation produced in one of the topic sentences used in PÁGICO might be:
  - Original sentence:

    `Filmes sobre a ditadura ou sobre o golpe militar no Brasil`[1]
  - Annotated sentence:

    `Filmes\N sobre\PRP a\ART ditadura\N ou\CONJ-C golpe\N militar\ADJ em\PRP o\ART Brasil\PROP`

- Chunking, where a set of rules for grouping tokens was used, based on their POS tags and relations between them — mostly sequence relations. After chunking,

[1] Translated into English, it reads as "Movies about the dictatorship or the military *coup* in Brazil."

the previous sentence would end up split as:

```
{Filmes}\NP sobre\ADVP {a ditadura}\NP ou sobre\ADVP {o golpe militar}\NP
em\ADVP {o Brasil}\NP
```

On POS tagging, it is worth noticing that some care was taken in the use of the *OpenNLP* POS tagger. For example, we sought to keep compound names (such as person names, countries, or places) bundled together and identified as a single element by the POS tagger, for better analysis and manipulation in the noun phrases. For such, all tokens in the sentences were previously processed, using named entity recognition (even though the classification was ignored, as we only intended to group compound names). For instance, it is more useful to classify "*Universidade de Coimbra*," with the tokens grouped, as `{Universidade de Coimbra}\PROP`, rather than as `Universidade\PROP de\PRP Coimbra\PROP`.[2]

The chunking process was not perfect. However, the identification of the chunks as NP or VP, based essentially on the presence of names and articles in the first, and of verb inflections on the second, was enough for the needs of the approach at the time. The rules used for chunking were extracted from *Bosque* (Afonso et al., 2002), after an analysis of how many times the same POS tags are found together in the same chunk, and of their sequence. Examples of rules used for aggregating tokens in noun phrases are presented in Fig. 4.2, depicting POS tag sequences (in the `lhs` attribute of the rules) and corresponding chunk types. After the splitting of questions in chunks, those became the cornerstone in the processing of questions.

```
<rule lhs="art; adj; n">np</rule>
<rule lhs="art; pron-det; adj; n">np</rule>
<rule lhs="art; pron-det; adj; prop">np</rule>
<rule lhs="art; pron-det; num; adj; n">np</rule>
<rule lhs="art; pron-det; num; n">np</rule>
<rule lhs="art; pron-det; prop">np</rule>
<rule lhs="pron-det; n">np</rule>
<rule lhs="pron-det; prop">np</rule>
<rule lhs="pron-indp; art; n">np</rule>
<rule lhs="pron-indp; n">np</rule>
```

Figure 4.2: RAPPORTÁGICO's chunker rules examples

4.3.2 Answer Category

As mentioned earlier, we have considered the first noun in the first NP of each topic to be the target of the topic — i.e., the noun is the category to which all possible answers have

[2] Actually, after applying NER, *Universidade de Coimbra* gets its tokens grouped by replacing the white spaces with underscores, before being processed by the POS tagger: `Universidade_de_Coimbra`.

to obey. In other words, the noun can be considered as a hypernym of the entities that will be given as answers, much like Ferreira et al. (2008) do to identify the category of entities, who also consider that the first sentence in the Wikipedia article usually defines the entity to which the article refers. For instance, using the topic "*Filmes sobre a ditadura ou sobre o golpe militar no Brasil*," which was the first topic of PÁGICO, we end up having `filme`[3] as the first (and only) noun of the first NP of that topic. As such, the target of the topic would be "movies" (`filmes`).

Although there are multiple patterns that express the relation of hypernymy in corpora, when the text consists of definitions, a recurrent pattern is *<hyponym> is a <hypernym>*. This happens because a common way of defining a concept is through the structure: proximal genus (*genus*), which is usually a hypernym, and difference (*differentia*). It is in this way that dictionary definitions are structured (see, for instance, Amsler (1981)). Likewise, the works of Snow et al. (2005) or Navigli and Velardi (2004), for English, and Freitas et al. (2008), for Portuguese, use this pattern. In the context of Wikipedia, the pattern *is a* has also been productive in the acquisition of hypernymy, as is the case of the works of Herbelot and Copestake (2006), for English, and Gonçalo Oliveira et al. (2011), for Portuguese.

As such, for building the query, we started by putting the previous pattern before the category. So, for example, if the target was `filme` (the noun in the first NP), the first part of the query would be `(é um filme) OR (são um filme) OR (foi um filme) OR (foram um filme)`. Notice that there was no concern in making number concordance, because after stemming it would be eventually ignored.

4.3.3 Synonym Expansion

In order to increase the search scope, it is possible to set alternatives to some words. In this case, the alternatives would be words with the same meaning — that is, synonyms. For setting those alternatives in the queries, we used the `OR` operator. Although it is possible, for instance, to get synonyms from any open category word, we have performed only experiments where we got synonyms of the noun that represents the topic's category, and also from VP constituted by just one verb.

For instance, the `músico`[4] category may have as synonyms the words `musicista` or `instrumentista`, that, in some contexts, have the same meaning. The same way, the verbs `escrever` and `utilizar`[5] may have as alternatives, respectively, the words `redigir` and `grafar`, and the words `usar` and `empregar`.

After verifying that the synonym expansion of the category increased the spreading

[3]Translated to English as "movie" or "film."
[4]Translated to English as "musician."
[5]Translated to English as "to write" and "to utilize," respectively.

of answers, we have chosen to use synonyms only for verbs.

As a synonym base, we used *synsets* from ONTO.PT, a lexical ontology for Portuguese, build automatically from lexical resources, and structured in a similar way to Princeton's Wordnet (Fellbaum, 1998). In the context of Wordnet, *synsets* are sets of synonyms which may be seen as the lexicalization of concepts from natural language. Ideally, a word will belong to a *synset* for each of its meanings, and words that, in a given context, may have the same meaning should be included, at least, in a same *synset*.

In the ONTO.PT version used in this work, the existing *synsets* consisted in the *synsets* of an electronic *thesaurus* of the Portuguese language, manually created, called TeP (da Silva and de Moraes, 2003). Prior to use, TeP was automatically enriched (Gonçalo Oliveira, 2013) with synonymy data from CARTÃO (Gonçalo Oliveira et al., 2011), which, by its turn, was extracted from three Portuguese electronic dictionaries.

As words with multiple senses may be included in more than one *synset*, getting synonyms is no trivial task, and implies the establishment of a correspondence between the word's occurrence and its closest sense. For such, it was necessary to use an algorithm for word sense disambiguation (see, for instance, Navigli (2009) for a review of techniques for this task), selecting the *synset* that corresponds to the topic's context. Two different algorithms were used, both based on exploring the structure of ONTO.PT, the *synsets*, and the relations between them, namely the *Bag-of-Words* method (using and adaptation of the Lesk algorithm (Lesk, 1986)), and the *Personalized PageRank* method (Brin and Page, 1998).

To prevent the query from becoming too large and include infrequent words, when *synsets* with too many elements are selected, only alternative synonymous with over twenty occurrences in the corpora of the AC/DC project (Santos and Bick, 2000) were used. For this purpose, we used word frequency lists from corpora made available through Linguateca.

4.3.4 Nationality or Country Expansion

Knowing in advance that the topics of PÁGICO would focus on Lusophone culture, an additional step was taken in the processing of topics, specifically dedicated to optimize the expansion of expressions related to the eight Portuguese-speaking countries and corresponding nationalities was included. This phase was divided into two parts:

- For each occurrence of a nationality of a Portuguese-speaking country, it was included in the query, as an alternative, the name of the country. For example, using the English translation, the processing of the phrase `Brazilian football`, gives rise to the alternative:
  `(Brazilian Football) OR (Football AND Brazil)`

- Every occurrence of expressions such as `Lusophone country`, `Portuguese` (language) or `former colony` was enriched with the name of each of the Lusophone countries. Thus, for example, to process the expression `Lusophone country`, the following restriction is obtained (using here the English translation, as it happened in the example above):
  ```
  (Portuguese-speaking country) OR Portugal OR Angola OR Brazil OR
  Mozambique OR (Cape Verde) OR (Guinea Bissau) OR
  (Sao Tome and Principe) OR Timor
  ```

It was sought, this way and in this case, to make related queries with this structure as wide as possible, without, however, leading to a loss of accuracy.

4.4 Index Searching

The third step consisted of searching the index for getting the most relevant articles matching the query produced in the previous stage. Each phase of the processing step generated one or more restrictions over the articles. The restrictions were combined in the query using the `AND` operator.

It was also defined that only Lucene results with a score above zero should be considered, and then ordered accordingly with their relevance, while ignoring those outside of the first n returned. For the official participation, we had empirically defined $n = 25$, as the answer threshold for all topics.

4.5 Answer Processing

After retrieving the articles that were considered relevant for the query, the system proceeded to remove articles whose types were previously known as not being answers, such as: structure-related pages (e.g., pages with names starting with *Wikipédia*, *Portal*, *Lista* ou *Anexo*); disambiguation pages; articles started with digits; articles referring to fields of knowledge (e.g., *Economia*, *Historiografia*, *Demografia*), and pages regarding things such as practices, conditions, principles, and doctrines, with names ending in "*ismo*" (e.g., *Anarquismo*, *Academicismo*, *Abolicionismo*). Although now we would do it differently, changing the order of the steps, at the time the application of this exclusion list was only done after removing the results outside the first n (25) returned.

For some of the articles that should be excluded from further processing, it is clear their exclusion, such as those related with Wikipedia's structure; others were excluded based on an analysis of the intended answer types and on the analysis of some recurring results, which never contained the expected answer. For instance, we had verified that

the answers for the test topics were always concrete cases, and not abstractions, such as disciplines, movements, principles, or ideologies.

Chapter 5

RAPPORT

In this chapter, we present all major aspects of RAPPORT, ranging from its division in four modules to each of its individual tasks and associated tools. This description includes how the RAPPORT system was developed and implemented, starting with an overview of the system's architecture, briefly describing each of the major modules that compose it, allowing a quick grasp of the key aspects of the approach. We proceed then to present a depiction of each of the modules (portraying their implementation and tools used, adapted or created) and of how they interact.

Special detail goes to the implementation of LEMPORT, a project that started as a way of having a high performing lemmatizer that could be easily integrated in RAPPORT, and ended up as standalone project on its own. LEMPORT is included in the description of the tools used and created in the scope of RAPPORT's implementation, namely on the fact extraction module.

5.1 Overview of the Approach

RAPPORT is an open domain system that mostly follows a typical framework for a QA system based on information retrieval, incorporating, at the same time, some characteristics of knowledge-based systems.

One of the most differentiating features of the approach is the use of facts as the basic unit of information regarding any entity found in a sentence. Facts are represented by triples, with subject, predicate and object, and other metadata elements, and are used as the basis for answering questions, as stated before. This approach shares also some similarities with open information extraction, regarding extraction and storage of information in triples (Gamallo, 2014).

RAPPORT depends on a combination of four modules, addressing information extraction, storage, querying and retrieving. The basic structure of the system can be seen in Fig. 5.1, and comprehends the following modules:

- fact extraction;
- fact storage;
- fact search;
- and answer retrieving.

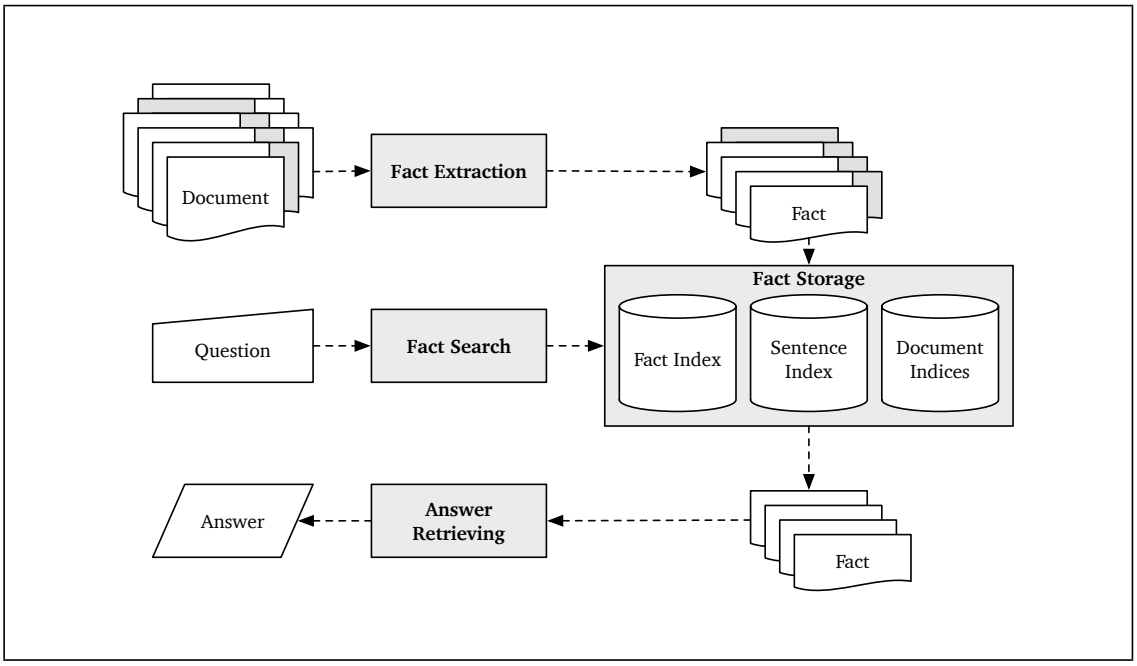

Figure 5.1: RAPPORT's general structure

The fact extraction and storage modules are performed offline, as soon as the corpus is available, and prior to the questions being posted in the system, while the fact search and answer retrieving modules are performed online.

Each of these modules is composed of multiple individual tasks that will be described in the next sections. A more detailed representation of the modules that compose RAPPORT can be observed in Fig. 5.2, identifying each of the tasks performed and implemented in them.

For the implementation of the modules, since the very beginning we have felt that there was no need to *reinvent the wheel* when good approaches to some NLP tasks already existed, and we could hardly do better in those specific domains, without diverting from our main goal. As such, we tried to use the most of existing NLP tools that would satisfy the requirements of RAPPORT. That is why we have resorted to some tools of the *Apache OpenNLP* toolkit, and also to *MaltParser*. It made sense to us using these tools, as there was little benefit in developing our own alternatives. We have created wrappers for them in order to ease their integration on our project, with some of them being improved by either pre-processing their input or post-processing their output.

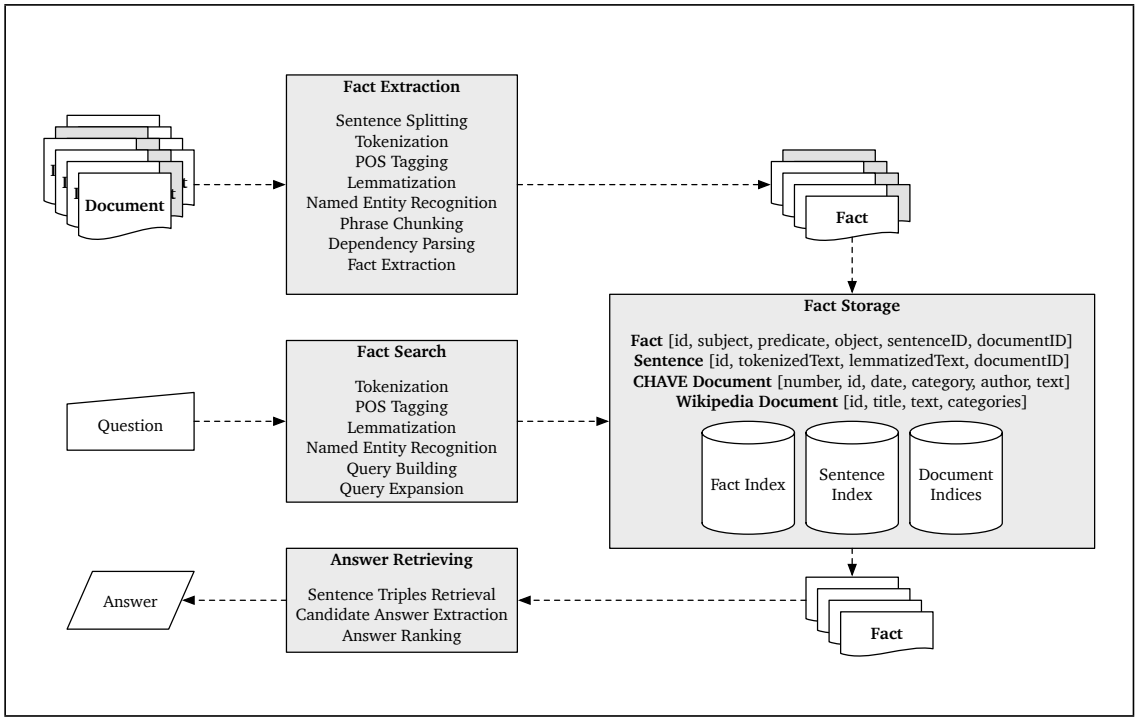

Figure 5.2: RAPPORT's detailed structure

The system starts by processing the corpus, document by document, and then sentence by sentence, in order to extract and store meaningful information. This information consists of short facts, with an *object* characterizing a *subject* through a *verb* (a predicate) — a triple, for all purposes. The facts are then stored in an index, alongside an index for sentences and another two for documents. The indices are all interrelated through the use of identifiers for each triple, sentence and document, allowing to cross from an index to another. These two modules are run offline, when the corpus that will support the QA system is defined.

Then come the online modules of the system: fact search and answer retrieving. The fact search module revolves around processing the users questions, identifying the key elements and then using to produce a query to search the indices, leading to the selection of the facts that will provide the answer. Once the query is performed and the facts retrieved, they are ordered and processed for presenting just the passage that answers the question (alongside the document identifier) in the answer retrieving module. Each of these modules is detailed in the next sections.

5.2 Fact Extraction

The system has formally its start in the fact extraction module. This first module is performed offline and is responsible for transforming natural language texts into a formal structure that can be more easily processed. It is in this module that are performed all

the tasks that allow the system to go from text to facts. There are eight individual tasks, including *fact extraction*, which lends its name to this module:

1. **Sentence splitting**: The texts are initially processed and split into sentences. For CHAVE, the texts correspond to each of the documents found in every edition of the newspapers in the corpus. For Wikipedia, a text is an article stripped of everything that is not actual text (i.e., it is removed the *wiki* related format and structure), with additional sentences added at the end in the guise of a `is a` template, using the title of the article and each of the categories it belongs to.

2. **Tokenization**: Each sentence is tokenized, in order to separate all words and punctuation marks (tokens). The tokens form the base of, or are used in, most of the remaining tasks.

3. **Named entity recognition**: The tokens are processed for recognizing named entities, such as persons, organizations, places, and dates. In addition to being used again later, the resulting entities are also fed back to the tokenizer in order to group together tokens that constitute a named entity, and to treat them as such.

4. **Part-of-speech tagging**: Each token is assigned the corresponding part-of-speech tag, identifying its syntactic function in the sentence.

5. **Lemmatization**: With both the tokens and respective POS tags, lemmatization is performed in order to retrieve the corresponding lemmas, allowing the system to use just the base form of a word.

6. **Phrase chunking**: Using tokens once again, and the respective POS tags, alongside with the lemmas, phrase chunking is performed yielding *phrase chunks* — including noun chunks, verb chunks, and adjective chunks.

7. **Dependency parsing**: Also using tokens, together with POS tags and lemmas, dependency parsing is done, yielding *dependency chunks* — chunks that are built using the dependencies between tokens, grouping them according to major dependencies, such as subjects, verbs, and objects.

8. **Fact extraction**: Finally, using named entities, proper nouns, phrase and dependencies chunks, fact extraction is performed.

In this first module, the corpora are processed, picking each of the documents, splitting sentences, identifying tokens, and extracting facts. It includes multiple tasks, namely sentence splitting, chunking, tokenization, POS tagging, lemmatization, dependency parsing, and NER.

An overview of the whole process is shown in Fig. 5.3, identifying all the tasks performed in the module and how they interact together.

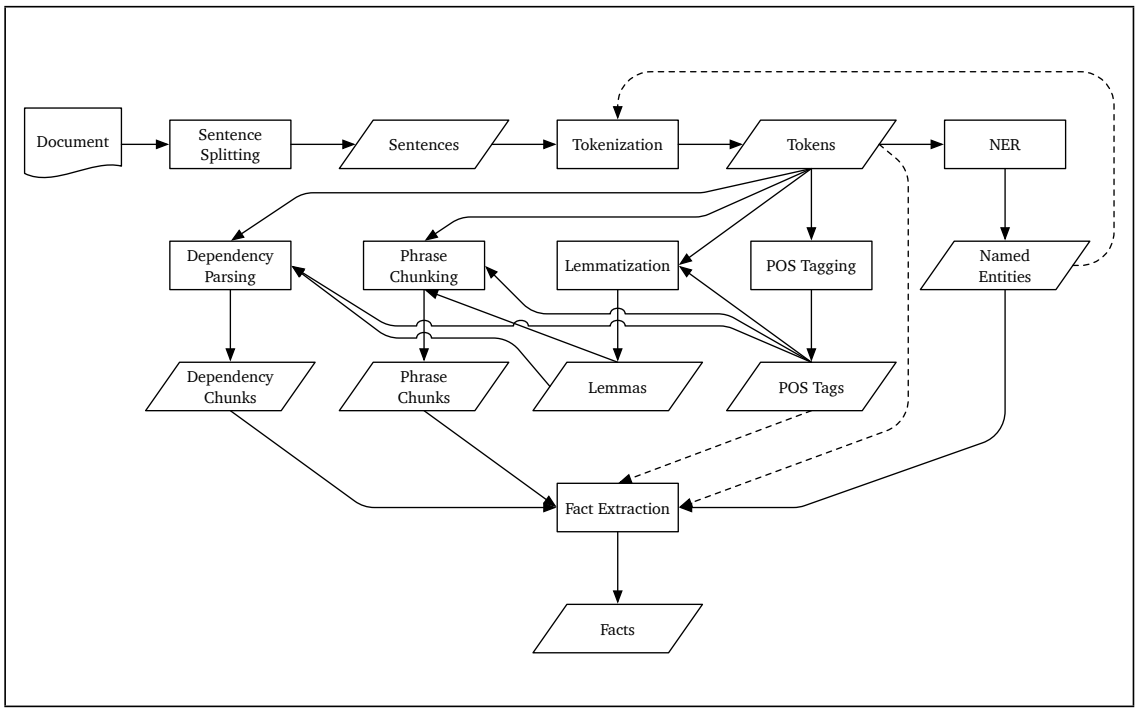

Figure 5.3: Fact extraction overview

The sentence splitting, tokenization, POS tagging, phrase chunking, and NER tasks are done resorting to tools included in the *Apache OpenNLP toolkit*, with minor tweaks whenever needed, such as grouping person names in a single token in the tokenization task, instead of using a token for each of the words, improving the handling of the tokens by the tools that follow.

For the lemmatization process, we have used LEMPORT. Another tool was used for dependency parsing, namely, *MaltParser*, with the model used by the parser being trained on the *Bosque 8.0* treebank. The output of *MaltParser* is further processed in order to group the tokens around the *main* dependencies, including: *subject*, *root* (*verb*), and *object*.

Specifically regarding fact extraction, it is performed using two complementary approaches, both involving named entities, or alternatively proper nouns, as a way of determining which facts are of future use. The facts are defined by three fields: *subject*, *predicate*, and *object*. After the documents are split into sentences, each sentence is directly processed in order to extract named entities. Then, the sentences either are chunked or undergo tokenization, POS tagging and lemmatization before applying *MaltParser* to identify the main dependencies. The algorithm that describes the process is found in Alg. 5.1.

As it can be noticed, only the facts with entities or proper nouns in the *subject* or

```
Data: corpus documents
Result: triples
read documents;
foreach document do
    split sentences;
    foreach sentence do
        tokenize, lemmatize, POS tag, chunk, and dependency parse;
        identify proper nouns;
        extract named entities;
        get phrase chunks;
        foreach phrase chunk do
            if phrase chunk contains any entity or proper noun then
                if adjacent phrase chunk has a specific type then
                    create fact relating both chunks, depending on the adjacent chunk type
                      and contents;
                end
            end
        end
        get dependency chunks;
        foreach chunk do
            if chunk contains any entity or proper noun and is a subject or an object then
                create fact using the subject or object, the root, and corresponding object or
                  subject, respectively;
            end
        end
    end
end
```

Algorithm 5.1: Overall fact extraction algorithm

in the *object* are later stored, and used for future querying — which may be seen as a limitation. The predicate has the verb stored in its lemmatized form in order to facilitate later matches against keywords found in the questions (namely, the verb). As for the other words, being entities or proper nouns, they are usually already in their *base* form.

In the facts that are based on the proximity between chunks, most of the predicates comprehend the verbs *ser* (to be), *pertencer* (to belong), *haver* (to have), and *ficar* (to be located). For example, if two NP chunks are found one after another, and the first chunk contains a named entity, it is highly probable that it is further characterized by the second chunk. If the second chunk starts with a determinant or a noun, the predicate of the future triple is set to *ser*; if it starts with the preposition *em* (in), it is used the verb *ficar*; if it starts with the preposition *de* (of), it is used the verb *pertencer*; and so on. Concretizing, in the sentence "*Mel Blanc, o homem que deu a sua voz a o coelho mais famoso de o mundo, Bugs Bunny, era alérgico a cenouras*" yields three (suitable) distinct facts: *{Bugs Bunny} {ser} {o coelho mais famoso do mundo}* and *{Mel Blanc} {ser} {o homem que deu a sua voz ao coelho mais famoso do mundo}*, both using the chunk proximity approach, and then using the dependency chunking approach, *{Mel Blanc} {ser} {alérgico a cenouras}*.

5.2.1 Sentence Splitting

One of the first tools used was the *Apache OpenNLP*'s sentence detector for splitting text documents into sentences. Usually, given a text, the first thing to do is to split it into sentences and process them individually, applying later other tools to each sentence, namely the tokenizer.

This tool was the basis of our sentence splitter, and we have used the *OpenNLP*'s `SentenceDetectorME` class[1] with only two major tweaks:

- A list of abbreviations was used in order to avoid splitting sentences on the periods that are common in abbreviations;
- We have also defined that line breaks would always result in a new sentence — as it was overwhelmingly the case with both corpora used (CHAVE and Wikipedia).

The tweak regarding the splitting of sentences at a line break was performed before applying *OpenNLP*'s sentence detector. This task was easily addressed, defining a regular expression for the purpose (depicted in Fig. 5.4).[2] Line breaks are used for splitting sentences because the texts in CHAVE and in Wikipedia, except for lists and titles, have no sentences spanning across line breaks.

```
<replacement target="(\n\r?)|(\r\n?)"></replacement>
```

Figure 5.4: Regular expression for splitting sentences on line breaks

As for the abbreviations tweak, we applied it following sentence splitting: whenever a sentence ends with an abbreviation, it is attached with the sentence following it, if any. In Fig. 5.5, it can be seen a few examples from the abbreviation list compiled and used in the sentence splitter.

Regarding the model, we used the one for Portuguese made available from *OpenNLP* (`pt-sent.bin`),[3] which is based on *CoNLL-X Bosque* data.

5.2.2 Tokenization

Another tool that was recurrently used was the *Apache OpenNLP*'s tokenizer, specifically the `TokenizerME` class, together with *OpenNLP*'s pre-trained model for Portuguese

[1] The `ME` suffix in some of *OpenNLP*'s classes denotes the use of a maximum entropy model.

[2] In this figure and in many of those that follow, it is shown excerpts of an XML based format, where all "rules" are defined by means of a *target*, used to find matches in text, and of an eventual *replacement*, that, depending on the task at hand, may be optional.

[3] This and other *Apache OpenNLP*'s pre-trained models used in this work can be freely downloaded from `http://opennlp.sourceforge.net/models-1.5/`.

```
<replacement target="q.b."></replacement>
<replacement target="q.e.d."></replacement>
<replacement target="q.e.d."></replacement>
<replacement target="q.g."></replacement>
<replacement target="q.-g."></replacement>
<replacement target="q.i.d."></replacement>
<replacement target="q.l."></replacement>
<replacement target="q.p."></replacement>
<replacement target="q.s."></replacement>
<replacement target="q.s.i.d."></replacement>
<replacement target="q.v."></replacement>
<replacement target="ql."></replacement>
```

Figure 5.5: Examples of abbreviations used in the sentence splitter

(`pt-token.bin`). The class was used also with some tweaks: contractions and clitics were considered in pre-processing the sentences for the tokenizer; and abbreviations were considered in the post-processing of the sentences.

After sentences are tokenized, the tokens are checked for the presence of contractions and clitics in them, in order to better address part-of-speech tagging later. Expanding clitics in tokens, separating clearly the verb and the personal pronouns, makes it easier to identify pronouns and verbs by the POS tagger. For example, using the rules shown in Fig. 5.6, we get from *dar-me-ia*,[4] that would be POS tagged just as a verb, to *daria a mim*, yielding as tags a verb, a preposition and a pronoun, respectively for the three resulting tokens.

```
<replacement target="-ma"> ela a mim</replacement>
<replacement target="-mas"> elas a mim</replacement>
<replacement target="-me-á">á a mim</replacement>
<replacement target="-me-ão">ão a mim</replacement>
<replacement target="-me-ás">ás a mim</replacement>
<replacement target="-me-ei">ei a mim</replacement>
<replacement target="-me-emos">emos a mim</replacement>
<replacement target="-me-ia">ia a mim</replacement>
<replacement target="-me"> a mim</replacement>
<replacement target="-mo"> ele a mim</replacement>
<replacement target="-mos"> eles a mim</replacement>
```

Figure 5.6: Examples of clitics processed by the tokenizer

Regarding contractions, the reason for processing them is similar to that of clitics, for instance, breaking apart prepositions and pronouns, as shown in Fig. 5.7. For example, *aos* (a preposition) is changed into *a os* (a preposition and a pronoun).

[4] In English, *"[it] would give me."*

```
<replacement target="à">a a</replacement>
<replacement target="ao">a o</replacement>
<replacement target="aos">a os</replacement>
<replacement target="àquela">a aquela</replacement>
<replacement target="àquelas">a aquelas</replacement>
<replacement target="àquele">a aquele</replacement>
<replacement target="àqueles">a aqueles</replacement>
<replacement target="aqueloutra">aquela outra</replacement>
<replacement target="àqueloutra">a aquela outra</replacement>
<replacement target="aqueloutras">aquelas outras</replacement>
<replacement target="àqueloutras">a aquelas outras</replacement>
<replacement target="aqueloutro">aquele outro</replacement>
<replacement target="àqueloutro">a aquele outro</replacement>
<replacement target="aqueloutros">aqueles outros</replacement>
<replacement target="àqueloutros">a aqueles outros</replacement>
<replacement target="àquilo">a aquilo</replacement>
<replacement target="às">a as</replacement>
```

Figure 5.7: Examples of contractions processed by the tokenizer

The abbreviation list is used for a reason similar to that on sentence splitting, getting the period coupled with the respective abbreviation (and classified together) instead of being addressed as punctuation and leading to incorrect classifications from the POS tagger. For abbreviation examples, please refer back to Fig. 5.5. Abbreviations with multiple periods in them that may have been split by the tokenizer are also put back together, as in *q. b.* back to *q.b.* ("as much as is needed," in culinary uses).

We have also opted for grouping tokens during the tokenization process: proper nouns were combined in a "unbreakable" token, to be processed together (feeding back the resulting entities from NER to the tokenizer); and adverbial expressions had their elements grouped together, with some examples presented in Fig. 5.8.

```
<replacement target="em abono de"></replacement>
<replacement target="em bloco"></replacement>
<replacement target="em breve"></replacement>
<replacement target="em carne e osso"></replacement>
<replacement target="em casa"></replacement>
<replacement target="em causa"></replacement>
<replacement target="em cima"></replacement>
<replacement target="em cima de"></replacement>
<replacement target="em conjunto"></replacement>
<replacement target="em conta"></replacement>
<replacement target="em contrapartida"></replacement>
<replacement target="em curso"></replacement>
```

Figure 5.8: Examples of token groups used in the tokenizer

5.2.3 Named Entity Recognition

The named entity recognizer was based on *OpenNLP*'s `NameFinderME` class, and was used straight out-of-the box. However, there was no pre-trained model for NER for Portuguese available among the *Apache OpenNLP* models. As such, a model had to be trained, for which we used the *Floresta Virgem* treebank in the format *árvores deitadas*. The trained model achieves a precision of 0.819, a recall of 0.768 and an *F*-measure of 0.793 over the used treebank.

Entities recognized by this tool, with the trained model, end up being classified as one of the following types: *abstract*, *artprod* (article or product), *event*, *numeric*, *organization*, *person*, *place*, *thing*, or *time*.

As stated before, the entities recognized by this tool are fed back to the tokenizer in order to bundle together the tokens that compose the entities, so that they can be identified and processed as such. This way, when the tokens that compose a multiword named entity get to the POS tagger, they get tagged together — for instance, as a proper noun in the case of the name of a person — instead of being individually tagged, with benefits in POS tagging process itself and later in other tasks that depend on it.

5.2.4 Part-Of-Speech Tagging

We used straight out-of-the-box the POS tagger found in the *OpenNLP* toolkit, given that the processing done previously in the tokenizer already addressed most of the issues that could affect the outcome of the tagging process. Specifically, we used the `POSTaggerME` class, using also the *OpenNLP*'s model for Portuguese (`pt-pos-maxent.bin`), with the option going once more for the ME model. Only a wrapper was created for ease of integration with the other RAPPORT classes. For reference, the POS tags for Portuguese available in the model are "*adjectivo*," "*advérbio*," "*artigo*," "*nome*," "*numeral*," "*nome próprio*," "*preposição*" and "*verbo*" — and, if considered as such, "*pontuação*."[5]

5.2.5 Lemmatization (LEMPORT)

For lemmatization, we have developed LEMPORT. We have opted for developing a lemmatizer from scratch as there was no ready to use lemmatizer that could be directly included in our project pipeline in a straightforward way, and because we believed that we could attain results in line with existing lemmatizers for Portuguese, if not better.

In the context of RAPPORT, we also believed that a lemmatizer would be a better approach than a simpler, but disruptive, stemmer. As stated earlier, in some situations,

[5] Corresponding to the English *adjective*, *adverb*, *article*, *noun*, *number*, *proper noun*, *preposition*, *verb*, and *punctuation*, respectively.

lemmatization and stemming operate in a similar way: given a set of affixes, for each word in a list (a phrase, a sentence or a text), check if the word ends with any of the affixes, and, if so, and apart from a few exceptions, remove the affix from the word. The problem is that this process is most of times not enough or even proper to retrieve the dictionary form of a word, it is too disruptive and, in most cases, the stem of a word is not the same as its lemma.

The Lemmatization Process

In order to retrieve the lemma of a Portuguese word, it is sometimes enough to remove the word's suffix. This typically happens in *noun number normalization*, which is the case of `carros` losing the trailing `s` and becoming `carro` (`cars` → `car`). In other cases, such as in *noun gender normalization*, it is necessary to replace the affix. For instance, the dictionary form of `gata` (a female cat) is `gato` (a male cat), which requires replacing the feminine affix, `a`, for the masculine affix, `o`. The same applies to *verb normalization*: for instance, the lemmatized version of `[eu] estudei` (`[I] studied`) becomes `[eu] estudar` (`[to] study`), replacing the verbal inflection affix with the associated conjugation infinitive affix (`ar`, `er`, or `ir`).

For Portuguese, lemmatization may include the following types of normalization for each part-of-speech: noun (gender, number, augmentative and diminutive), adjective (gender, number, augmentative, diminutive, and superlative), article (gender and number), pronoun (gender and number), preposition (gender and number), adverb (manner) and verb (regular and irregular). Proper nouns, numbers, interjections and conjunctions are ignored, as they are not normally inflected in Portuguese.

Determining what kind of normalization to apply depends on the syntactic category, or part-of-speech (POS) tag, of each word. The task of identifying the POS tag of a word is performed earlier by the already described POS tagger. So, a lemmatizer must take as input both word and POS tag to produce the coveted lemma.

In theory, knowing the syntactic category of a word and the rules to normalize it, lemmatization should be a straightforward process. In practice, although these rules cover the vast majority of the cases in any given text, exceptions to these rules defeat the goal of reaching an accuracy close to 100%. Moreover, the exceptions usually happen to be found in the oldest and most used lexemes of any lexicon. For instance, the verbs *to be* (*ser*) and *to have* (*ter*) are highly irregular in every western language, including English and Portuguese.

The same happens in other syntactic categories, such as nouns. For instance, the singular form of `capitães` (`captains`) is `capitão`, and not `capitãe`$*$,[6] obtained by

[6]The affix of a trailing asterisk ($*$) to a word denotes an *invalid* word.

solely removing the trailing `s`, the most common rule, which fails in some words due to their specific etymology. There are also cases where the masculine form of a noun is quite different from the feminine version (where a simple affix replacement ends up failing). One of the most problematic case is perhaps when a word is already in its dictionary form but appears to be in an inflected form, when, actually, it is not — as it happens with `farinha` (`flour`), and `sardinha` (`sardine`), that may seem to be diminutives, due to their ending in `inha`, the most common feminine diminutive affix.

Most lemmatization tools use a rule system (covering the vast majority of cases for each type of word normalization), specifying also exceptions to those rules, using, at some point, a lexicon for validating the lemmas produced, or for extracting rules (and their exceptions) based on that lexicon.

Related Work

Even though most researchers on Portuguese NLP must use some sort of lemmatization tool, those tools are not easily available, at least as isolated lemmatizers, although there are suites of tools, including morphological analyzers, that produce lemmas as a part of their outcome. Unfortunately, these suites and morphological analyzers leave to the users the task of post-processing the output in order to extract just the intended lemmas. Moreover, these tools also impose restrictions on the pre-processing tasks, as they usually have to be done using tools of the same suite.

Three such tools are known to exist, targeting specifically the Portuguese language: jSpell,[7] FreeLing,[8] and LX-Suite.[9] All have web interfaces, with jSpell and Freeling providing downloadable versions and source code. Moreover, jSpell is available as `C` and Perl libraries, as well as a MS Windows binary; and Freeling is available as a Debian package and also as a MS Windows binary, in addition to an API in Java and another in Python, along with the native `C++` API.

Both jSpell and Freeling start with a collection of lemmas and use rules to create inflections and derivations from those lemmas, alongside data such as number, syntactic category, or person, in the case of verbs (Simões and Almeida, 2001; Carreras et al., 2004). It is the output of that process that is used to lemmatize words, matching them against the produced inflections and derivations, retrieving the originating lemma.

Though only a description of a nominal lemmatizer (Branco and Silva, 2006) has been found, which we believe to be the lemmatizer supporting the LX-Suite (Branco and Silva, 2007) of NLP tools (belonging to the LX-Center), it uses general lemmatization

[7] jSpell is freely available from `http://natura.di.uminho.pt/wiki/doku.php?id=ferramentas:jspell` [Accessed: February 2017].

[8] Freeling is freely available from `http://nlp.lsi.upc.edu/freeling/` [Accessed: February 2017].

[9] The LX-Suite can be found at `http://lxcenter.di.fc.ul.pt/services/en/LXServicesSuite.html` [Accessed: February 2017].

rules and a lexicon for the purpose of retrieving exceptions to rules. If the lexicon contains a word (therefore, a valid word) that could be processed by the rules, but should not, it is marked as an exception and added to the exception list. According to the authors, that lemmatizer achieved an accuracy of 97.87%, when tested against a hand annotated corpus with 260,000 tokens produced by the same authors.

For jSpell and Freeling, although evaluations for the morphological analyzers do exist, no statement regarding specifically the accuracy of the respective lemmatizers was found.

Our Approach to Lemmatization

Our lemmatization method shares features with other approaches, such as the use of rules and, later, a lexicon. The way these resources are combined leads to a high accuracy value, above 98%. Next, we describe our method and its evolution.

The Use of Rules

Each of the normalization steps included in the lemmatization process had an associated set of rules. As such, there were rules for (and in this order):

1. manner (adverb) normalization;
2. number normalization;
3. superlative normalization;
4. augmentative normalization;
5. diminutive normalization;
6. gender normalization;
7. verb normalization (for both regular and irregular verbs).

Each of these rules were associated to one or more POS tags. So, for instance, gender normalization could be applied to nouns and also to adjectives, and articles, while superlative normalization would only be applied to adjectives. The rules were defined by the target affix, the POS tags of the words they should be applied to, exceptions, and the replacement for the target affix. All rules were declared in XML files, illustrated by the example in Fig. 5.9. That specific rule would transform, for example, `malinha` (small briefcase) into `mala`, by replacing the affix `inha` for the affix `a`, but would leave `farinha` untouched, as it is one of that rule's exceptions.

Although the rules for all types of normalization shared the same generic structure, the rules regarding verbs were somewhat specific:

```
<prefix>[\wàáãâéêíóõôúç\-]*</prefix>
...
<replacement target="inha" tag="n|n-adj|adj" →
exceptions="azinha|...|farinha|...|sardinha|...|vizinha">a</replacement>
```

Figure 5.9: A rule for transforming a diminutive into its *normal* form.

(i) The rules for lemmatizing irregular verbs consisted of all the possible inflections of the Portuguese irregular verbs. It was simpler (and more precise) to do this than to come up with rules that could address all existing variations.

(ii) The rules for the regular verbs used as target the stem of each verb, always ending in a consonant, followed by the only vocals that could be appended to that specific stem (depending on the conjugation the verb belongs to), ending with any sequence of letters. Small variations that may occur — for instance, the substitution of a `g` for a `j` in the verb `agir` (to act) in some of its inflections — were also considered.

(iii) When the two previous types of rules failed to be applied, a set of rules with verbal inflection affixes was used.

We have resorted to regular expressions for specifying the normalization rules. Any of these rules can accept a list of prefixes, to broaden the list of addressed verbs. An example of rules for regular verbs is shown in Fig. 5.10, where it is also shown the list of prefixes that can be added to a verb, and the ending (suffix) of all the verb rules.

```
<prefix>(a|ab|abs|...|sub|super|supra|...|vis).?\-?</prefix>
<suffix>[\wàáãâéêíóõôúç\-]*</sufix>
...
<replacement target="afunil[aáeo]" tag="v|v-fin|v-ger|v-pcp|v-inf"> →
afunilar</replacement>
<replacement target="a[gj][aeiío]" tag="v|v-fin|v-ger|v-pcp|v-inf"> →
agir</replacement>
```

Figure 5.10: A rule for transforming a inflected verb into its infinitive form

When more than one rule was eligible for application to a word in a given step, the lengthier one was chosen. The lengthier rules are typically more specific, including exceptions to the rules, while the shorter ones tend to be more generic, being used when the more specific fail. The length of a rule is computed by a weighted sum of the number of characters in the target, the exceptions and the POS tags, ordering the rules from a higher to a lower weight, as seen in Fig. 5.11, with the target being the most prominent element in a rule.

```
rule length = length of target × 1000 + length of exceptions + length of
replacement + length of prefix + length of tag + length of suffix
```

Figure 5.11: Formula for calculating the length of rules

The rules were mainly the result of our knowledge of the Portuguese language, the use of specialized literature, such as "*Nova Gramática do Português Contemporâneo*," by Cunha and Cintra (2002), and the analysis of the output of the lemmatizer, identifying and correcting errors. Exceptions were obtained in the same way, and added manually, resulting from the utilization of the lemmatizer (mainly from the analysis of errors).

It is worth noticing that the rules are easily readable and customizable. Moreover, it is possible to select which kind of normalization steps should be performed, by specifying flags on the calls to the lemmatizer — when none is specified, it defaults to apply all the normalization steps. Both of these features make our lemmatizer flexible and easy to adapt to different situations and purposes.

The Addition of a Lexicon

The current version builds up on the previous (using rules), with the addition of a lexicon, namely the "LABEL-LEX-sw" lexicon,[10] version 4.1, produced by LabEL (Eleutério et al., 2003). This lexicon contains over 1,500,000 inflected forms, automatically generated from about 120,000 lemmas, characterized by morphology and category attributes.

Beyond using this lexicon for validating the lemmas produced by the lemmatizer, we have used the fact that each entry of the lexicon contains the inflected form, lemma, syntactic category, syntactic subcategory, and morphological attributes, that can be directly applied in the lemmatization process. Fig. 5.12 shows an example of these entries regarding the word `gato` (cat) and its declensions.

```
gata,gato.N+z1:fs                gatita,gato.N+z1:Dfs
gatas,gato.N+z1:fp               gatitas,gato.N+z1:Dfp
gatinha,gato.N+z1:Dfs            gatito,gato.N+z1:Dms
gatinhas,gato.N+z1:Dfp           gatitos,gato.N+z1:Dmp
gatinho,gato.N+z1:Dms            gato,gato.N+z1:ms
gatinhos,gato.N+z1:Dmp           gatos,gato.N+z1:mp
```

Figure 5.12: An example of LABEL-LEX-sw lexicon entries

Using this lexicon provided an easy way of retrieving the lemma of any word, given its syntactic category. Also, the rules previously defined are now used only in a fallback

[10] The LABEL-LEX-sw lexicon is provided by LabEL, through `http://label.ist.utl.pt` [Accessed: February 2017].

scenario, when a word is not found in the lexicon. With one advantage: virtually all exceptions to the rules are already present in the lexicon, so that the probability of a rule failing is extremely low. This comes from the already stated fact that exceptions are usually found in extremely frequent, ancient, and well known words of a lexicon, rather than in more recent (and more regular), less used, or obscure words.

However, this does not mean that the lexicon could be used right out-of-the-box. Some issues had to be properly addressed: (*i*) mapping the syntactic categories present in the lexicon to the ones used on the rest of the program (including the rules used in earlier versions); (*ii*) excluding all pronoun and determiner lexicon entries, as they present disputable normalization — for instance, `tu` (you) has `eu` (I) as its lemma; and (*iii*) making optional some gender normalizations, such as presenting `homem` (man) as the lemma of `mulher` (woman).

When a word is shared by multiple lemmas, the lemma with the highest frequency is selected. For this purpose, we have used the frequency list of the combined lemmas present in all the Portuguese corpora available through Linguateca's AC/DC project (Santos and Bick, 2000).[11]

The only perceived drawback of the method is the time it may take, even if only a couple of seconds, to load the lexicon into memory. Other than that, it is quite performant, as the lexicon is stored in a hash structure, which is known to be a fast method for storing and searching on sets of elements. A lemma cache is also used, with each word that is found in the analyzed text to be stored in memory together with is syntactic category (POS tag) and lemma, at run-time, which avoids searching again the whole lexicon or selecting which rule to apply for a word processed previously. The cache is a distinctive improvement to performance speed because, additionally to a set of words commonly used across different domains, texts on a specific topic tend to have their own set of words that are used time and again. The basic structure of the used lemmatization algorithm is presented in Alg. 5.2.

Regarding flexibilization, additionally to the customization of rules and selection of which normalization steps to apply, the lemmatizer allows the option to add new entries to a custom lexicon (if it fits best to do so in a lexicon, instead of specifying an exception in the rules, or both).

Evaluation and Results

For the lemmatizer evaluation, we have used *Bosque 8.0* treebank, which was parsed in order to retrieve, for each word found in it, the inflected form, its syntactic category, and corresponding lemma. The inflected form and syntactic category were fed to

[11] The word frequency lists of AC/DC are provided by Linguateca, through `http://www.linguateca.pt/ACDC` [Accessed: February 2017].

```
Data: token, tag
Result: lemma
create/use cache;
load/use lexicon;
load/use rules;
if cache contains (token, tag) then
    lemma = cache (token, tag);
end
else if lexicon contains (token, tag) then
    lemma = lexicon (token, tag);
    add (token, tag, lemma) to cache;
end
else
    foreach rule in rules [adverb, number, superlative, augmentative, diminutive, gender, and
     verb] do
        lemma = normalize (lemma, tag, rule);
        if lexicon contains (lemma, tag) then
            add (token, tag, lemma) to cache;
            stop;
        end
        else
            continue;
        end
    end
end
```

Algorithm 5.2: Lemmatization algorithm

our lemmatizer, and its output was matched against the known lemma, as originally identified in *Bosque*.

In Table 5.1 we can see the overall results using rules alone, the lexicon alone, and both rules and lexicon (the current version of the lemmatizer), applied to the Portuguese and Brazilian parts of *Bosque*, with the current version reaching an accuracy value over 98%. The same table also presents the results broken down into three major syntactic categories: nouns, adjectives, and verbs.

Table 5.1: Overall and partial results in major categories.

Bosque	Only Rules	Only Lexicon	Both Rules and Lexicon
Overall PT	97.76%	95.06%	98.62%
Overall BR	97.67%	95.16%	98.56%
Nouns PT	96.94%	98.05%	98.30%
Nouns BR	96.40%	96.67%	97.86%
Adjectives PT	90.10%	95.39%	98.19%
Adjectives BR	88.77%	91.70%	97.23%
Verbs PT	98.04%	88.78%	98.79%
Verbs BR	98.34%	89.59%	99.15%

Using both lexicon and rules significantly improves the normalization of adjectives against using only rules. The same happens in the normalization of other parts-of-speech, even if to a lower degree. The lexicon alone does not cover all the cases either,

as it is virtually impossible for a lexicon, comprehensive as it may be, and even by its very nature, to cover all the lexemes, and associated syntactic categories, in any language. For instance, past participles are not contemplated in the used lexicon in their plural inflections (which is common case when used as adjectives). That may be one of the why reasons rules perform better than the lexicon on verbs, in addition to possessing a more extensive verb list.

A brief analysis of a random sample of 10% (160) of the cases where the lemmatizer produced different lemmas on the Portuguese part of *Bosque* is presented in Table 5.2. Detailing the different types of errors and discrepancies: *incorrect categorization* refers to a token being incorrectly tagged in Bosque, affecting the output of the lemmatizer which uses the same tag; *both lemmas acceptable* happens when Bosque has one lemma for the token and the lemmatizer produces another, while both are acceptable; *orthographic errors* refer to an error in the Bosque token, affecting the output of the lemmatizer, which tries to lemmatize the error; LEMPORT *errors* refer to errors in the output of the lemmatizer, failing to produce the right lemma; and *Bosque errors* refer to errors in the lemmas presented in Bosque. Of these errors and discrepancies, only LEMPORT *errors* can be directly attributable to the lemmatizer, while the others affect its output even when there is no fault in the lemmatizer.

Table 5.2: Errors and discrepancies identified in both LEMPORT and Bosque

Type	Quantity	Example (*Form#POS:Bosque:*LEMPORT)
Incorrect categorization	1.25%	`Afeganistão#N:afeganistão:afeganisto*`
Both lemmas acceptable	2.50%	`cabine#N:cabine:cabina`
Orthographic errors	3.75%	`ecxemplares*#N:exemplar:ecxemplar*`
LEMPORT errors	43.75%	`presas#V-PCP:prender:presar*`
Bosque lemmatization errors	48.75%	`pais#N:pais*:pai`

The accuracy could be even higher (probably slightly above 99%), as in a significant amount of the faulty cases the problem may actually be found in the *Bosque* annotation (lemma and POS tag) for each of the tokens in a sentence.

Assessment and Integration

LEMPORT, through the conjugation of simple rules with a comprehensive lexicon, is able to achieve a high overall accuracy, over 98%, making it suitable for use in many NLP tasks for Portuguese. In addition to RAPPORT, versions of the lemmatizer have been used in a question generation system (Diéguez et al., 2011), in the creation of the CARTÃO lexical-semantic resources (Gonçalo Oliveira et al., 2011), in question answering for multiple choice questions (Gonçalo Oliveira et al., 2014), in the analysis of news titles (Gonçalo Oliveira et al., 2016), in semantic alignment and similarity (Oliveira Alves et al., 2016; Gonçalo Oliveira et al., 2017), and Onto.PT (Gonçalo Oliveira and

Gomes, 2010), and even in other works abroad, such as in mapping verbs to knowledge base relations (Wijaya and Mitchell, 2016). We believe that this adoption of the lemmatizer denotes its usefulness and quality.

Although the margin for improvement is narrow, there is still room to improve the lemmatizer by addressing some minor but troublesome issues, such as compound and hyphenated words, as well as multiword expressions. One of these issues is already partially tackled by splitting the words at the hyphen, sharing the syntactic function between them. However, there are cases where elements of composed and hyphenated words, when put apart, belong to different categories.

Other issues may include the processing of oblique cases in pronouns. The *Bosque* corpus presents the oblique case and the pronoun that would be the corresponding lemma, but we usually process the oblique cases prior in the tokenization process.

Notwithstanding, LEMPORT is the lemmatizer used in RAPPORT, being perhaps the most constant element of the whole approach, in its multiple iterations and evolution during its development.

5.2.6 Phrase Chunking

The chunker was also used out-of-the-box, using the class `ChunkerME`. However, as also happened in NER, no prebuilt model for Portuguese was available. In this case, we have used once more the *Bosque 8.0* treebank, for both training and for testing the model, yielding an accuracy of 0.95, a recall of 0.96, and an F-measure of 0.95. The chunker has as input the tokens, and their grammatical tags, as well as the lemmas. The chunks can be classified as nominal, verbal or prepositional.

Again, with the exception of small aspects related with the presentation of results, including in the description of the chunks also the lemmas (which are not considered in the original version of *Apache OpenNLP* chunker), the results were used directly in the system.

5.2.7 Dependency Parsing

For dependency parsing, we have resorted to *MaltParser*.[12] We use the dependencies as a way of aggregating tokens from a sentence in chunks, instead of using just the dependencies *per se*. For example, we would want to group all the tokens related to the noun identified as the subject of the sentence, rather than just get the noun by itself.

The model for *MaltParser* was also trained using Linguateca's *Bosque 8.0*, after conversion to the CoNLL-X format. Resulting from the application of that model, a token can be assigned one of many grammatical functions. Of those grammatical functions,

[12] *MaltParser*'s website can be found at `http://www.maltparser.org/` [Accessed: February 2017].

only the following can be selected as direct dependents of the *root* token in the next processing step (except for the *root* token itself and *punctuation* tokens):

- (Predicate) Auxiliary Verb (`PAUX`);
- (Predicate) Main Verb (`PMV`);
- Adjunct Adverbial (`ADVL`);
- Adjunct Predicative (`PRED`);
- Auxiliary Verb (`AUX`);
- Complementizer Dependent (`>S`);
- Dative Object (`DAT`);
- Direct Object (`ACC`);
- Focus Marker (`FOC`);
- Main Verb (`MV`);
- Object Complement (`OC`);
- Object Related Argument Adverbial (`ADVO`);
- Passive Adjunct (`PASS`);
- Predicator (`P`);
- Prepositional Object (`PIV`);
- Punctuation (`PUNC`);
- Root (`ROOT`);
- Statement Predicative (`S<`);
- Subject (`SUBJ`);
- Subject Complement (`SC`);
- Subject Related Argument Adverbial (`ADVS`);
- Top Node Noun Phrase (`NPHR`);
- Topic Constituent (`TOP`);
- Vocative Adjunct (`VOC`).

After the sentence is processed by the dependency parser, the tokens are grouped in what we have chosen to call *dependency chunks*. The dependency chunks are formed by selecting the tokens whose head is the *root* of the sentence, and then by aggregating each of those tokens together with all their dependents. Please refer to Fig. 5.13 for an example of *dependency chunking*.[13] Furthermore, an algorithm describing the *dependency chunking* process can be seen in Alg. 5.3.

5.2.8 Fact Extraction

For fact extraction, we have built a special purpose tool that makes use of named entities and of phrase chunks or dependency chunks. Facts are, for all purposes, triples with extra information or metadata. Regardless of the corpus used (CHAVE or Wikipedia), a fact is composed of:

[13] For the sake of clarity, in the example given, some of the CoNNL-X fields in the lines resulting from the dependency parser are stripped off. The same happens to some of the contents of the chunks.

```
[tokens]
id          form          lemma         pos          head         dependency
 1          Mel_Blanc     mel_blanc     prop         21           SUBJ
 2          ,             ,             punc         1            PUNC
 3          o             o             art          4            >N
 4          homem         homem         n            1            N<PRED
 5          que           que           pron-indp    6            SUBJ
 6          deu           dar           v-fin        4            N<
 7          a             o             art          9            >N
 8          sua           seu           pron-det     9            >N
 9          voz           voz           n            6            ACC
10          a             a             prp          6            PIV
11          o             o             art          12           >N
12          coelho        n             n            10           P<
13          mais          mais          adv          12           N<
14          famoso        adj           adj          13           H
15          de            de            prp          12           N<
16          o             o             art          17           >N
17          mundo         mundo         n            15           P<
18          ,             ,             punc         12           PUNC
19          Bugs_Bunny    bugs_bunny    prop         12           APP
20          ,             ,             punc         12           PUNC
21          era           ser           v-fin        0            ROOT
22          alérgico      adj           adj          21           SC
23          a             a             prp          22           A<
24          cenouras      cenoura       n            23           P<
25          .             .             punc         21           PUNC

↦

[chunks]
id          head          function      tokens
 1          2             SUBJ          [ Mel_Blanc o homem que deu a sua voz a
                                           o coelho mais famoso de o mundo Bugs_Bunny ]
 2          0             ROOT          [ era ]
 3          2             SC            [ alérgico a cenouras ]
```

Figure 5.13: Dependency parsing and chunking example

- an **identifier** — a unique identifier of a fact in relation to each sentence, auto-incremented;
- a **subject** — the subject of the fact, usually a named entity or some thing related to the object;
- a **predicate** — the predicate of the fact, typically a verb;
- an **object** — the object of the fact, usually some thing related to the subject or a named entity, reverse mirroring the contents of the subject;
- a **sentence identifier** — a unique identifier of a sentence in relation to each document, auto-incremented;
- a **document identifier** — a unique identifier, the `DOCID` in the case of CHAVE, or the article filename in the case of Wikipedia.

The rules for extracting facts from sentences are only used in the case of adjacent *phrase chunks*. For *dependency chunks*, although it can also be considered as some kind

```
Data: tokens, tags, lemmas, listOfMajorDependencies
Result: chunks
dependencyTokens = parse (tokens, tags, lemmas);
foreach dependencyToken in dependencyTokens do
    if dependencyToken is root then
        chunk = dependencyToken;
        add chunk to chunks;
    end
    else if dependencyToken is dependent of root then
        chunk = dependencyToken;
        chunk += getDependents of dependencyToken;
        add chunk to chunks;
    end
end
sort chunks based on the id of the main dependency of each chunk;
```

Algorithm 5.3: Dependency chunking algorithm

of rule, we use directly the dependency classification of the chunks, as in `SUBJ + ROOT + OBJ` → `subject + predicate + object`. In Fig. 5.14, it is shown some of such rules for extracting facts of adjacent phrase chunks.

```
<replacement target="[NP][NP]">is a</replacement>
<replacement target="[NP][em:PP]">is in</replacement>
<replacement target="[NP][de:PP]">is part of</replacement>
```

Figure 5.14: Examples of rules for extracting facts of adjacent phrase chunks

When using phrase chunks, the fact extractor checks them for the presence of named entities. When a match occurs, adjacency relations between the chunks are used to extract facts. For instance, if a noun phrase chunk contains an entity of the type person and is immediately followed by another noun phrase chunk, it is highly probable that the second chunk is a definition of specification of the first, thus yielding the fact NP_n *is a* NP_{n+1}. The rules specify the classification of the adjacent chunks and also some elements that the chunks may or must contain, appended using a prefix or a suffix with a colon (`:`).

For dependency chunks, a different set of rules is used. In that situation, the *subject* and *object* chunks are checked for the presence of entities, and when a match occurs, a triple is built using the corresponding predicates and objects or subjects, accordingly. Then the subject, predicate, and object chunks are transposed into the corresponding fields of a newly created fact.

As can be noticed, fact extraction is currently limited to the presence of entities (or proper nouns, used when no entities are available or recognized as such) in both the phrase and dependency chunks. It does not have to be like that, but it is a form of

reducing the extraction of spurious facts. We do intend to revise this option in the near future, devising a way of selecting facts that may be meaningful even without named entities or proper nouns in them.

In Fig. 5.15, we present examples of facts extracted from a sentence (including an erroneous fact identified with an asterisk). In spite of the possibility of a few incorrect facts being extracted, it can be easily noticed that facts do summarize the key information bits present in the sentence. An overview of the fact extraction process is depicted in Alg. 5.4.

```
Mel_Blanc, o homem que deu a sua voz a o coelho mais famoso de o mundo,
Bugs_Bunny, era alérgico a cenouras.

↦

Fact:  {subject=[Bugs_Bunny], predicate=[ser], object=[a sua voz a o coelho mais
famoso de o mundo]}*
+
Fact:  {subject=[Mel_Blanc o homem que deu a sua voz a o coelho mais famoso de o
mundo Bugs_Bunny], predicate=[ser], object=[alérgico a cenouras]}
+
Fact:  {subject=[Mel_Blanc], predicate=[ser], object=[o homem que deu a sua voz a
o coelho mais famoso de o mundo Bugs_Bunny]}
+
Fact:  {[subject=[Mel_Blanc], predicate=[ser], object=[o homem]}
+
Fact:  {[subject=[Mel_Blanc], predicate=[ser], object=[alérgico a cenouras]}
```

Figure 5.15: Example of facts extracted from a sentence

Data: chunks, rules
Result: facts
foreach *chunk in chunks* **do**
  if *type of chunk is phrase chunk* **then**
    foreach *rule in rules* **do**
      if *rule applies to chunk* **then**
        create fact with chunk and adjacent chunk, using rule;
        add fact to facts;
      end
    end
  end
  else if *type of chunk is dependency chunk* **then**
    create fact with the chunk's subject, predicate and object;
    add fact to facts;
  end
end

Algorithm 5.4: Fact extraction algorithm

5.3 Fact Storage

Once the facts are extracted, they are stored alongside the sentences where they can be found, and the documents those sentences belong to. For that purpose, four distinct Lucene indices were created, with two of them depending on the corpora used:

- one fact index for storing the triples (subject, predicate and object), their *id*, and the *ids* of the sentences and documents that contain them;
- one sentence index for storing the sentence *ids* (a sequential number representing its order within the document), the tokenized text, the lemmatized text and the *ids* of the documents they belong to;
- two document indices store the data describing the document, as found in CHAVE (number, id, date, category and author), or in Wikipedia (by extracting *id*,[14] title, text and categories).

Although each index is, in practice, independent from the others, they can refer to one another by using the *ids* of the sentences and of the documents. That way, it is easy to determine the relations between documents, sentences, and triples. These indices, mainly the sentence and the fact indices, are then used in the next steps of the RAPPORT system, for querying and retrieving data. Even though there are two document indices, depending on the corpus being used (CHAVE or Wikipedia), when the distinction is not needed, we will refer only to a document index.

Lucene was used for storing the facts for one major reason: Lucene allows for partial and fuzzy matches, additionally to scoring each of the results, while yielding greater flexibility in the storage and retrieval of data. For achieving the same results, at least regarding partial and fuzzy matches, a triple store, such as *Sesame*,[15] would require complex SPARQL (Pérez et al., 2009) query constructs that would be heavily dependent on regular expressions.

The key elements of each fact are the subject, the predicate and the object, with the *id* being used for identifying univocally the triple, and the with sentence identifier and the document identifier to relate triples with the sentences they were extracted from and also the documents — as stated earlier in the caracterization of facts.

The sentence index stores for each sentence: an *id*, the tokenized text of the sentence and also a lemmatized version of the text, alongside a document identifier. The lemmatized text is used for later matching against the question queries, as the sentence index is the entry point in the query stage of the system.

[14] In practice, the filename of the article.
[15] For Sesame, currently *RDF4J*, please refer to `http://www.rdf4j.org/` [Accessed: February 2017].

Finally, the document index mimics either the data of a CHAVE news article or of a Wikipedia article, storing an *id*, category and text, for both cases. Depending on the document coming from CHAVE or Wikipedia, it also stores a number, a date and an author, or an URL, respectively. In practice, at least in the QA@CLEF benchmarks used for 2007 and 2008, two indices are used: one for CHAVE and another for Wikipedia. For QA@CLEF tracks, we used a snapshot from November 2006 (the same date used in the the original evaluation).

Only the contents of the sentence index are effectively lemmatized, together with a non-lemmatized version (for display purposes), as the queries are run against this index. The fact index is only used after, with the selection of the answers being made using mostly the named entities, that do not need to be lemmatized. The document index also does not need to be lemmatized, as the documents serve the only purpose of storing the whole text of a document.

In all the indices, the fields are stored as *strings*, with the exception of the text in the document index, which is stored as *text*. Fig. 5.16 depicts the structure of the used indices, and Fig. 5.17 shows index entries for each of the four indices.

```
Document Index [CHAVE]
NUMBER: string field, stored
ID: string field, stored
DATE: string field, stored
CATEGORY: string field, stored
AUTHOR: string field, stored
TEXT: text field, stored

Document Index [Wikipedia]
ID: string field, stored
TITLE: string field, stored
TEXT: text field, stored
CATEGORIES: string field, stored

Sentence Index
ID: string field, stored
TOKENIZED_TEXT: text field, stored
LEMMATIZED_TEXT: text field, stored
DOCUMENT_ID: string field, stored

Fact Index
ID: string field, stored
SUBJECT: string field, stored
PREDICATE: string field, stored
OBJECT: string field, stored
SENTENCE_ID: string field, stored
DOCUMENT_ID: string field, stored
```

Figure 5.16: Structure of the used indices

The indices are created and populated using the following sequence: fact index, sentence index, and document index. As such, for every document processed, an index entry is created for that document, the document text is split into sentences, which are also processed and stored in the corresponding index, to which follows fact extraction

```
Document Index [CHAVE]
[...]
NUMBER: PUBLICO-19950722-157
ID: PUBLICO-19950722-157
DATE: 19950722
CATEGORY: Sociedade
AUTHOR:
TEXT: [...]  Mel Blanc, o homem que deu a sua voz ao coelho mais famoso do mundo, Bugs Bunny, era
alérgico a cenouras.  No Alasca, é ilegal olhar para os alces da janela de um avião.  O rei francês
Carlos VI, convencido de que era feito de vidro, odiava viajar de coche porque as vibrações podiam
estilhaçar o seu nobre corpo...  Para todos aqueles que gostam de pormenores estranhos e fora do
vulgar, o Guiness Book of Records vai agora passar a ter um cantinho dedicado a todos estes detalhes
bizarros que dão cor os nosso estranho mundo.  [...]
[...]

Sentence Index
[...]
ID: 12
TOKENIZED_TEXT: Mel_Blanc , o homem que deu a sua voz a o coelho mais famoso de o mundo , Bugs_Bunny,
era alérgico a cenouras .
LEMMATIZED_TEXT: Mel_Blanc , o homem que dar o seu voz a o coelho mais famoso de o mundo ,
Bugs_Bunny, ser alérgico a cenoura .
DOCUMENT_ID: PUBLICO-19950722-157
[...]

Fact Index
[...]
ID: 6
SUBJECT: Mel_Blanc
PREDICATE: ser
OBJECT: alérgico a cenouras
SENTENCE_ID: 12
DOCUMENT_ID: PUBLICO-19950722-157
[...]
```

Figure 5.17: Index entries data examples

and subsequent storage in the fact index. An overview of the process is depicted in Alg. 5.5. However, entries in the sentence index are only executed if facts are found in a sentence, and the same applies to entries in the document index: document index entries depend on the documents having sentences with facts extracted.

5.4 Fact Search

The fact search module has some similarities with the fact extraction module, at the text processing level, being composed of six individual tasks: tokenization, POS tagging, lemmatization, NER, query building, and query expansion. More details on the query building and query expansion tasks are described next — the remaining tasks have already been addressed in Section 5.2, about fact extraction, earlier in this same chapter.

In this module, the questions go through the same processing that was applied earlier to sentences, removing stopwords and punctuation. All remaining words are used to build the query to be run against the sentence index. After the initial processing of the questions, this module first addresses query building and then query expansion.

```
Data: documents
Result: indices
foreach document in documents do
    split document into sentences;
    foreach sentence in sentences do
        extract facts from sentence;
        foreach fact in facts do
            store fact data in fact index;
        end
        if facts length > 0 then
            store sentence data in sentence index;
        end
    end
    if facts length > 0 then
        store document data in document index;
    end
end
```

Algorithm 5.5: Fact storage algorithm

The reason for using the sentence index and not directly the fact index is as follows: facts contain unlemmatized subsets of the tokens in a sentence, which makes them less suitable than sentences for querying; and sentences have more tokens to be matched against a query, in addition to them being lemmatized. Whenever a sentence matches a query, the associated facts and document data are retrieved — and this goes for all the sentences matching that query.

In a similar way to the annotation made to the sentences in the corpus, the questions are processed in order to extract tokens, lemmas and named entities, and identify their types and categories.

The query is then run against the sentence index — the system searches for sentences with the lemmas previously identified. When a match occurs, the associated facts are retrieved, along with the document data.

The facts that are related to the sentence are then processed, checking for the presence of the question's entities in either the *subject* or the *object* of the triples, for selecting which facts are of interest.

5.4.1 Query Building

For building the queries, the system starts by performing NER, POS tagging and lemmatization on the questions. The lemmas are useful for broadening the matches and results that could be found only by using directly the tokens. The queries are essentially made up of the lemmas found in the questions (including named entities and proper nouns).

In the queries, all elements are, by default, optional, except for named entities. If no entities are present in the questions, proper nouns are made mandatory keywords;

alternatively, if there are also no proper nouns in the questions, it is the nouns that are used as mandatory keywords in the queries. For instance, in order to retrieve the answer to the question "*A que era alérgico Mel Blanc?*,"[16] the query will end up being defined by `+Mel_Blanc ser alérgico`. We have opted for keeping all the lemmas, except for very common stop words, because Lucene scores higher the hits with the optional lemmas, and ends up ignoring them if they are not present.

Whenever there are no recognized named entities in the questions, proper nouns are used and set as mandatory. Eventually, if also no proper nouns are to be found in the questions, common nouns are used. A description of the query building process is found in Alg. 5.6.

```
Data: question
Result: query
tokenize question;
postag tokens;
lemmatize tokens;
remove stopwords from tokens;
if question contains named entities then
    foreach entity in named entities do
        query += "+" + named entity;
    end
    foreach token in tokens do
        if query does not contain token then
            query += token;
        end
    end
end
else if tokens contains proper nouns then
    foreach proper noun in tokens do
        query += "+" + proper noun;
    end
    foreach token in tokens do
        if query does not contain token then
            query += token;
        end
    end
end
else if tokens contains nouns then
    foreach noun in tokens do
        query += "+" + noun;
    end
    foreach token in tokens do
        if query does not contain token then
            query += token;
        end
    end
end
```

Algorithm 5.6: Query building algorithm

[16] In English, "What was Mel Blanc allergic to?"

5.4.2 Query Expansion

Query expansion is used in particular cases, namely for verbs (using synonyms) and for toponyms and demonyms, replacing, for instance, countries for nationalities and vice-versa. In the case of toponym expansion, it is only applied to named entities — as in replacing *Portugal* for *Portuguese*. For all expansion related to toponyms and demonyms, we have resorted data from the *Dicionário de Gentílicos e Topónimos*, available at the *Portal da Língua Portuguesa* website,[17] creating a double list of correspondences between toponyms and demonyms.

In this step, we also use a *nominalizer*, that for any verb found in a sentence applies a set of suffixes for creating a noun out of the verb.

As for other terms expansion, we have used ONTO.PT, extracting elements from *synsets* were the term to be expanded can be found. An illustration of the query expansion process is shown in Alg. 5.7.

```
Data: query
Result: expanded query
expanded query = query;
foreach keyword in query do
    if keyword has synonyms then
        add synonyms of the keyword to the query;
    end
    if keyword has toponyms then
        add toponyms of the keyword to the query;
    end
    if keyword has demonyms then
        add demonyms of the keyword to the query;
    end
    if keyword is a verb then
        add nouns resulting from the verb to the query;
    end
end
```

Algorithm 5.7: Query expansion algorithm

As we will see in the Chapter 6, Subsection 6.2.5, query expansion provides mixed results, with more right answers, but introducing with them also incorrect ones.

5.4.3 Sentence Facts Retrieval

After the creation of the query, the system uses it to match it against the sentence index, getting the results ordered by the associated Lucene hit scores. Then, the retrieval of the sentence facts is essentially a question of selecting all the facts present in the fact

[17] This resource can be retrieved from `http://www.portaldalinguaportuguesa.org/index.php?action=gentilicos&act=list` [Accessed: February 2017].

index that have a sentence identifier that matches that of each of the sentences retrieved after the question query matching.

For that to happen, for each of the sentences retrieved in the question query matching, it is used its *id* for selecting the corresponding facts in the fact index. All facts whose *sentence id* is a match to that of the sentence, are retrieved for further processing the next step.

An example of this process is presented in Fig. 5.18, where, for the query we have used previously, a sentence is retrieved as the best match, and, using its *id*, the related facts are also retrieved.

```
Query
+Mel_Blanc ser alérgico

Highest Sentence Match Score
ID: 12
TOKENIZED_TEXT: Mel_Blanc , o homem que deu a sua voz a o coelho mais famoso de o mundo , Bugs_Bunny,
era alérgico a cenouras .
LEMMATIZED_TEXT: Mel_Blanc , o homem que dar o seu voz a o coelho mais famoso de o mundo ,
Bugs_Bunny, ser alérgico a cenoura .
DOCUMENT_ID: PUBLICO-19950722-157

Sentence Related Facts
ID: 2
SUBJECT: Bugs_Bunny*
PREDICATE: ser*
OBJECT: a sua voz a o coelho mais famoso de o mundo*
SENTENCE_ID: 12
DOCUMENT_ID: PUBLICO-19950722-157

ID: 3
SUBJECT: Mel_Blanc o homem que deu a sua voz a o coelho mais famoso de o mundo Bugs_Bunny*
PREDICATE: ser
OBJECT: alérgico a cenouras
SENTENCE_ID: 12
DOCUMENT_ID: PUBLICO-19950722-157

ID: 4
SUBJECT: Mel_Blanc
PREDICATE: ser
OBJECT: o homem*
SENTENCE_ID: 12
DOCUMENT_ID: PUBLICO-19950722-157

ID: 5
SUBJECT: Mel_Blanc
PREDICATE: ser
OBJECT: o homem que deu a sua voz a o coelho mais famoso de o mundo Bugs_Bunny
SENTENCE_ID: 12
DOCUMENT_ID: PUBLICO-19950722-157

ID: 6
SUBJECT: Mel_Blanc
PREDICATE: ser
OBJECT: alérgico a cenouras
SENTENCE_ID: 12
DOCUMENT_ID: PUBLICO-19950722-157
```

Figure 5.18: Fact retrieval example

5.5 Answer Retrieving

As stated before, after a sentence matches a query, the associated facts and document data are retrieved. As the document data consists essentially of metadata — currently, it is only used for better characterizing the answers —, let us focus on the facts.

Here, the named entities found in the questions are used again to select which, of the selected facts should be considered. Just the facts with the named entity (or one of the entities) present in the question are kept. Then, for each fact, an answer snippet is extracted: if the best match against the query is found in the subject, the object is returned as being the answer; otherwise, if the best match is found against the object, the subject is returned.

This snippet, before being the answer to be presented later on to the user, may be ordered against other candidate answers. For that, the facts are used again, as the answer snippets are ordered against the number of facts they are found in. As it happens with query expansion, also the ordering method provides mixed results. By default, the facts assume an order based on the Lucene hit score of the associated sentences.

Finally, a last filter is applied: answers with entity types matching that of the questions are presented first, and then only the others. An algorithm describing both the data querying and this process is found in Alg. 5.8. For the QA@CLEF tracks, the types of the questions are found together with the question text, alongside the category and other meta data. For other questions, we resort to patterns and the presence of a list of words in the questions to help identify the types.

The correspondence between the QA@CLEF question types and entity types is as follows:

- `COUNT`: `numeric`;
- `LOCATION`: `place`;
- `MANNER`: *none*;
- `MEASURE`: `numeric`;
- `OBJECT`: `thing, artprod`;
- `ORGANIZATION`: `organization`;
- `PERSON`: `person`;
- `TIME`: `time`.

```
Data: question
Result: answers
create query using named entities, proper nouns, or nouns as mandatory, and the remaining
  lemmas from the question as optional;
run query against sentence index;
foreach sentence hit do
    retrieve facts related to the sentence hit;
    foreach fact in facts do
        if subject contains named entities from question then
            add object to answers and retrieve sentence and document associated with the
              fact;
        end
        else if object contains named entities from question then
            add subject to answers and retrieve sentence and document associated with the
              fact;
        end
    end
    order answers based in the number of triples they are found in;
    foreach answer in answers do
        if answer contains named entities with a type related to that of the question then
            add answer to primary list;
        end
        else
            add answer to secondary list;
        end
    end
    put primary list at the top of answers and the secondary list at the bottom;
end
```

Algorithm 5.8: Answer retrieval algorithm

Continuing with the example used, given the correct sentence is retrieved, of the related facts, the one that best matches the query is *{Mel Blanc} {ser} {alérgico a cenouras}*. Removing from the fact the known terms from que question, what remains must yield the answer: *[a] cenouras*. Besides that, as the named entity, Mel Blanc, is found in the subject of the triple, the answer is most likely to be found in the object, which is the presented as the answer.

5.6 Applying the Approach

In this section, we provide a sneak peek of how RAPPORT works using both CHAVE and Wikipedia. We focus on corpora processing, storing data in indices, user question processing, and question delivery to the user.

5.6.1 Corpora Processing

The first big step of the approach is to process the corpus or corpora used. While the CHAVE documents are easier to process (please refer back to Fig. 3.1, on page 36, for an

example of a CHAVE document), Wikipedia articles are more difficult, as, even though they follow is a predefined general structure, there are many variations within that structure (including the presence or absence of structural elements).

For processing CHAVE documents, since their structure is always the same, the only field that is effectively processed for later answer retrieving are `TEXT` (the actual contents of the news article). The other fields just store the respective data as they come from the documents. Nevertheless, the `DATE` field deserves a special mention, as some questions are date related and that information may not be explicit the contents). As the processing of CHAVE documents is straightforward, we will address next just the processing of Wikipedia articles.

The first issue with Wikipedia articles is to determine the main contents boundaries. For that purpose, we have performed essentially web scraping. In the snapshot used, the article's actual contents were delimited between the tags `<!-- start content -->` and `<!-- end content -->`, with some extra parsing for ignoring items such as the links for versions of the articles in other languages, categories, and other minor structural elements. Everything outside those tags is excluded from further processing.

The categories, while not part of the text of the articles, were nevertheless used to classify the articles in an *`is a`* manner, stating that *[article] name `is a` category*. This step is useful as a considerable number of questions are definitions and some of them may be answered by retrieving the categories to which an article is related to.

Although both CHAVE and Wikipedia are used in the same context and for the same purpose, a faithful characterization of both types of documents implies the use of two different indices. As such, there is an index for CHAVE documents and another for Wikipedia documents. There are however just one index for facts and one for sentences.

5.6.2 Question Processing

All CLEF questions were compiled from various sources and stored in one single file just with those targeted at the Portuguese language. Questions are processed one at a time, using the questions' text to create the query that will be used against the sentence index and to retrieve associated facts. Then, there is an option to expand the terms of the query. Besides that, the type of the questions is also defined for helping sort the candidate answers. In addition to the use of the question's type, the sentences are ordered against Lucene's hit score.

5.6.3 Answer Delivery

Finally, each fact is processed in order to extract the intended answer. The answers for each question are presented ordered based on the number of occurrences of the answers

in all the retrieved facts, or resorting to the hit score of the sentences that contained the facts, filtered by the question type, and then matched against the known answers, classifying them as right (if there is a match) or wrong (if there is no match). For a match to be considered, it is enough that the text is essentially the same, even if it is found in a document not in the known results.

Chapter 6

Evaluation

In this chapter we describe the experimental work done to test and evaluate RAPPORT. We also address how RAPPORTÁGICO performed in PÁGICO, with the first section dealing with that competition and results achieved. Then, we proceed to describe how the system performs with the QA@CLEF tracks from 2004 to 2008, using Portuguese, comparing it against the other known approaches, and how it has changed between iterations. We also discuss some of the different configurations tested.

6.1 PÁGICO

Our participation in the PÁGICO evaluation context consisted of three official runs, using RAPPORTÁGICO. In addition to sharing the indexed corpus, the runs had the following in common while processing the topics: they resorted to chunking and used both noun chunks and verb chunks, when available, as elements of the restrictions; they identified the category and used the *is a* pattern; and they expanded countries and nationalities in the queries.

The differences among the three runs were as follows, all of them regarding topic processing:

1. The first, which could be seen as a *baseline*, with no synonym expansion.

2. The second performed synonym expansion in the verb phrases containing just a verb, using the method *Bag-of-Words* for disambiguation of terms.

3. The third was almost identical to the second, but used the method *Personalized PageRank* for the disambiguation of terms.

In addition to the three official runs, two more runs were sent outside the official period. In these two runs, the category, obtained from the noun in the first NP, was also

expanded using synonyms, additionally to the VP expansion. Each of these runs also uses one of two methods of disambiguation, mirroring the second and third runs.

We have assessed that for each verb that was subjected to synonym expansion, on average, 11.6 and 6.5 synonyms were obtained, in the second and third run, respectively. Regarding the expansion of the categories, we obtained, on average, 5.9 and 6.4 synonyms for each, respectively in the fourth and fifth runs — the unofficial runs.

It should be noted that after the submission of the runs, we have found that there were problems in the code for disambiguation, which had prevented the context from being taken into account. Thus, in the five runs herein described, choosing the best *synset* was actually made as follows: in the runs using the *Bag-of-Words* algorithm, it was chosen a random *synset*; the remaining runs used the simple *PageRank* algorithm, instead of *Personalized PageRank*. That is, the *synset* with the best score was always chosen, given the structure of the graph, and regardless of any context. Nonetheless, specially in words with little ambiguity, this situation should not have affected the results significantly, although this is one of the things we intend to verify in a possible new iteration of RAPPORTAGICO.

6.1.1 Results

Recalling what has been previously stated in Section 3.4, and shown in Table 3.16 (back in page 60), in the PÁGICO evaluation, there were seven participants, which, of those, five were humans and two were computer systems. Of the humans, two of the participants were actually groups of people, with each element addressing subsets of the 150 topics (*ludIT* and *GLNISTT*), and the other three were individuals. Much probably due to the amount of work required, just the participants that were teams were able to answer close to the total 150 topics.

Assessing the official results against those from our approach: from a total of 12 submissions allocated among the various participants, RAPPORTÁGICO got the fifth, sixth and seventh places with scores of 25.0081, 23.7379, and 19.0693 for runs 3, 2 and 1, respectively. It was, nevertheless, the best of the two computational approaches and better than one of the human contestants.

Table 6.1 summarizes of the results of our approach, presenting the total number of correct answers ($|C|$), and the number of topics which received at least one right answer in each of the runs, with different cut-off points. It also shows the results for various cut-off points (limits) on the number of answers submitted by topic ($|R|$), precision ($|P|$), pseudo-recall (α) and the corresponding score (M). Keep in mind that the official results refer to a cut-off point of a maximum of 25 answers by topic, as mentioned previously, identified in bold; the italics are the best partial results for each run.

Table 6.1: Results of the multiple runs

Run	Limit	$\|C\|$	$\|R\|$	$\|P\|$	α	M	# Topics
	5	86	512	*0.1680*	0.0383	14.4453	47
	10	122	918	0.1329	0.0543	16.2135	51
1	15	147	1275	0.1153	0.0654	16.9482	54
	20	164	1577	0.1040	0.0730	17.0551	56
	25	***181***	**1718**	**0.1054**	***0.0805***	***19.0693***	***59***
	5	90	516	*0.1744*	0.0400	15.6977	50
	10	132	927	0.1424	0.0587	18.7961	53
2	15	164	1289	0.1272	0.0730	18.3986	58
	20	184	1591	0.1157	0.0819	21.2797	58
	25	***203***	**1736**	**0.1169**	***0.0903***	***23.7379***	***59***
	5	92	518	*0.1776*	0.0409	16.3398	48
	10	135	940	0.1436	0.0601	19.3883	53
3	15	166	1305	0.1272	0.0738	21.1157	57
	20	188	1601	0.1174	0.0836	22.0762	58
	25	***208***	**1730**	**0.1202**	***0.0925***	***25.0081***	***59***

It is possible to observe the existence of a certain proportionality in the results of the various cut-off points, since they increase the number of answers submitted, the number of correct answers, and the number of topics with at least one right answer. So does the score for each of the limits. This can be an indicator that the cut-off point initially defined could be slightly higher — however, this analysis only came afterwards, and when we submitted the results we were not yet sure how the approach would be evaluated. Notice also that for the lower cut-off points of 5, 10 and 15, although there are fewer correct answers, the accuracy is higher than those of the official runs, given the more favorable ratio between the number of correct answers and the total number of answers submitted for evaluation. That is to say that, by changing the cut-off points, we are indirectly adjusting the robustness of the answer retrieving algorithm.

Although it was not possible to compare exhaustively the presence or absence of specific responses in each of the runs, by means of a simple analysis of differences in the output lines of each run, containing the answers, we were able to assert that the more diverse in terms of results were the second and third runs, with 123 different answers. The number of different answers between the first and second runs was 78, and between the first and the third runs was 101. Although only a small portion of the different answers are correct answers, each run gets a small set of answers that are not shared with the others.

There is a noteworthy aspect: all of the runs provided correct answers to the same number of topics. This shows that the various runs essentially differ in the number of correct answers for each topic presented. One hypothesis is that the terms found in the *initial* question (or topic) are the ones that best define the desired answers. Everything else in processing the topics essentially helps you find more alternatives (both correct and incorrect) answers.

Another interesting aspect is the fact that the restrictions of the topics are often found distributed by the contents of the articles, across several sentences, or even paragraphs, which suggests that all text is important for obtaining answers to complex questions — contrary to the sometimes recurrent belief that the most important element in getting a answer is the discovery of a specific phrase (with or without variations).

Concerning the two unofficial runs, we have achieved a somewhat surprising outcome: contrary to what could be expected, the number of generated answers had apparently decreased (to 1529 and 1519, respectively), and also the number of questions with answers had decreased (to 49 and 56, respectively). After analysis, we concluded that the number of generated answers in these two runs had not diminished; however, many of the (new) answers obtained with expansion of NP later came to be ignored because they were in the exclusion list. Furthermore, this exclusion list was only applied after obtaining the answers by Lucene. This leads us to believe that both the expansion of noun phrases and and verb phrases contribute to a wider scope of the *queries* but the expanded verb phrases end up being closer to the original meaning than the extended noun phrases. Regarding the score of these last two runs, reflecting the numbers of answers, the fourth run got a score of 16.1210, and the fifth, 19.7031. These runs also got a precision of 0.1027 and 0.1139 and a pseudo-recall of 0.0698 and 0.0769, respectively. Given these values, we deemed unproductive to investigate variations with different cut-off points.

6.2 QA@CLEF Tracks

In this section, we address the evaluation of RAPPORT. The system was tested and benchmarked using the questions (and known answers and results) from the QA@CLEF tracks for Portuguese from 2004 to 2008. We have opted for dividing the tests in three different sets, mainly according to the corpora used: first, just with the *Público* portion of CHAVE (2004); second, with the full CHAVE corpus, including *Público* and *Folha de São Paulo* (2005 and 2006); and then using both CHAVE and the Portuguese Wikipedia (2007 and 2008), ending with an overall appreciation.

The QA@CLEF tracks were perhaps the most evident way of coming up with a benchmark that we could use for the evaluation of RAPPORT, as both question and answers are known and accessible (although there may be answers outside of the official list, even if it is an extensive list), and the performance of the competitor systems in each of the tracks is also available.

The questions used in QA@CLEF adhere to the following criteria (Magnini et al., 2005): they can be *list* questions, *embedded* questions, *yes*/*no* questions (although none was found in the questions used for Portuguese), *who*, *what*, *where*, *when*, *why*, and *how*

questions, and definitions. In practice, for Portuguese, the questions were essentially divided in three categories (factoid, definition, and list) and eight types (person, organization, location, object, time, count, measure, and other).

For verifying if the answers presented by RAPPORT match the expected answers, the answers must match the already known answers. As a matter of fact, this validation may penalize our approach, as RAPPORT may find answers that, although not present in the list of known answers, may nevertheless be correct. In the evaluations in the next subsections, the system does not perform query expansion and the order of the facts is defined by the score of the sentences from which they were extracted. We are also considering right answers that match overall those known.

6.2.1 Evaluation using only *Público*

We start by focusing on the year of 2004, where the *Público* portion of CHAVE was the only corpus used for Portuguese, in order to have the same base corpus the other systems had at the time.

For reference, in Table 6.2 there is a summary of the best results for the Portuguese QA@CLEF tracks in 2004 (abridged from Magnini et al. (2005)), for multiple answer limits per question — in the official track, the limit was one answer per question. The results for our system are presented in the last row of the table.

Table 6.2: Comparison of the results at QA@CLEF 2004

Approach	Accuracy for multiple answer limits per question (%)						
	1	**5**	**10**	**20**	**25**	**50**	**100**
Esfinge	15.08	—	—	—	—	—	—
Senso	28.54	—	—	—	—	—	—
RAPPORT	17.59	36.18	41.21	47.24	50.25	54.77	61.31

In 2004, one of the questions was unintentionally duplicated. So, instead of 200, 199 questions were used for testing our system on that edition, of which 10% do not have an answer that can be found in the corpus — being 'NIL' the expected answer in that situation.

RAPPORT was able to find the answers to 15.59% of the questions (35 in 199), with a limit of one answer for each question. Using a limit of 10 answers per question, the system is able to find the right answer for 82 questions (41.21%). If that limit set to 100, the number of answered questions rises to 122, which may lead to the conclusion that one of the big issues to be further addressed is to improve the ordering and selection of the candidate answers, as the system seems to be able to produce sort answers, as intended. This assumption can be also drawn from the graph in Fig. 6.1, noticing the curve for the different cut-off points. Another conclusion is that the system, in the limit,

is only able to present the right answer to less than 70% of the questions, demonstrating how hard is to extract exact answer passages.

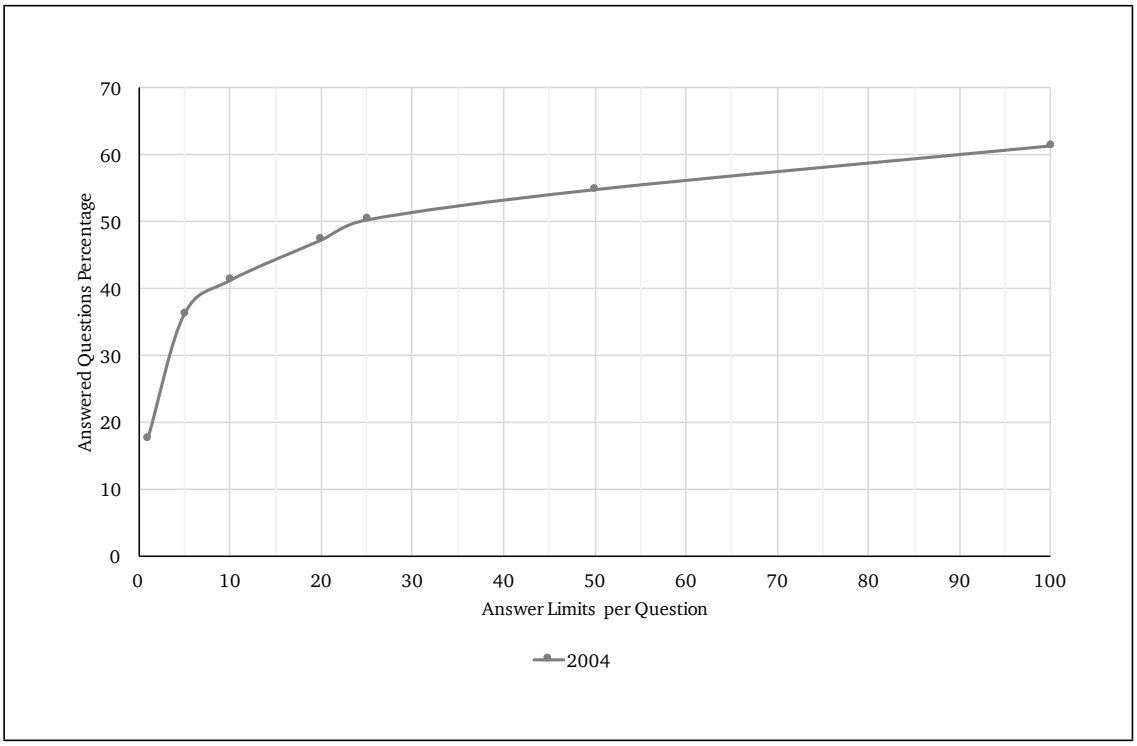

Figure 6.1: Retrieved answers with different limits for QA@CLEF 2004

Regarding the categories of the questions, for a limit of one answer per question, RAPPORT was able to answer correctly to 20.24% of the factoid questions, and to 3.23% of the definition questions. As for the results by question type, RAPPORT, answered correctly to 39.02% of location, 0% of manner, 4.35% of measure, 0% of object, 21.43% of organization, 0% of other, 13.60% of person, and 26.66% of time questions.

Finally, the system was able to present no answer to 25% of the questions whose expected answer was nil (that is, questions without answer in the corpus).

6.2.2 Evaluation using CHAVE

We now change the focus to the years of 2005 and 2006, where CHAVE (including both *Público* and *Folha de São Paulo*) was the corpus used for Portuguese. In these two years, in addition to definition and factoid questions, there were also list questions.

For reference, in Table 6.3 there is a summary of the best results for the Portuguese QA@CLEF tracks for 2005 (abridged from Vallin et al. (2006)), and in Table 6.4, the results for 2006 (abridged from Magnini et al. (2007)). At the end of both tables, the results of our system are also shown for different answer limits per question.

As stated before, we are only addressing here the questions for Portuguese used in the QA@CLEF tracks in 2005 and 2006. As such, a grand total of 400 questions (200

Table 6.3: Comparison of the results at QA@CLEF 2005

Approach	Accuracy for multiple answer limits per question (%)						
	1	5	10	20	25	50	100
Esfinge	23.00	—	—	—	—	—	—
Senso	25.00	—	—	—	—	—	—
Priberam's	64.50	—	—	—	—	—	—
RAPPORT	28.50	48.00	56.50	62.50	63.00	67.50	69.50

Table 6.4: Comparison of the results at QA@CLEF 2006

Approach	Accuracy for multiple answer limits per question (%)						
	1	5	10	20	25	50	100
Esfinge	24.50	—	23.56	—	—	—	—
Priberam's	67.00	—	67.00	—	—	—	—
NILC	1.50	—	1.50	—	—	—	—
RAPOSA	13.00	—	18.83	—	—	—	—
RAPPORT	18.00	32.50	38.00	43.00	44.00	50.00	54.00

in each year) were used for testing our system in those two editions, of which 10% do not have an answer that can be found in the corpus.

Using the set of questions from 2005 and 2006, and combining the results from both years, which were known to have their answers, if any, found on CHAVE, we were able to find the answers to 23.50% of the questions (93 in 400), grouping all the question from the those two editions of QA@CLEF, with a limit of one answer for each question. When that limit is lifted to 10 answers per question, the system is able to answer 189 questions (47.25%). When the answer limit is set to 100, 247 of the questions are correctly answered.

While our system is behind Priberam's in both years, in 2005 it was in front of Esfinge, in a notable increase in the number of correctly answered questions in that year, when compared to the previous year. It also continues to improve when the number of answers allowed increase. This is specially interesting in 2006, when the organization officially sanctioned also a limit of 10 answers per question. In that specific scenario, RAPPORT is able to get to the second place, only behind Priberam's system, answering correctly to 38% of the questions. The official results (also shown in Table 6.4) for that scenario are somewhat unusual: if for RAPOSA, as expected, the number of correctly answered questions increases, for the other three systems the numbers are the same or decrease marginally, as is the case of Priberam's. We were not able to assert the reason for such strange behavior in Magnini et al. (2007), which describes in detail the results of each system.

A graphic representation of the results of our system for different cut-off points in those two years is presented in Fig. 6.2.

Once more regarding the categories of the questions, for a limit of one answer per

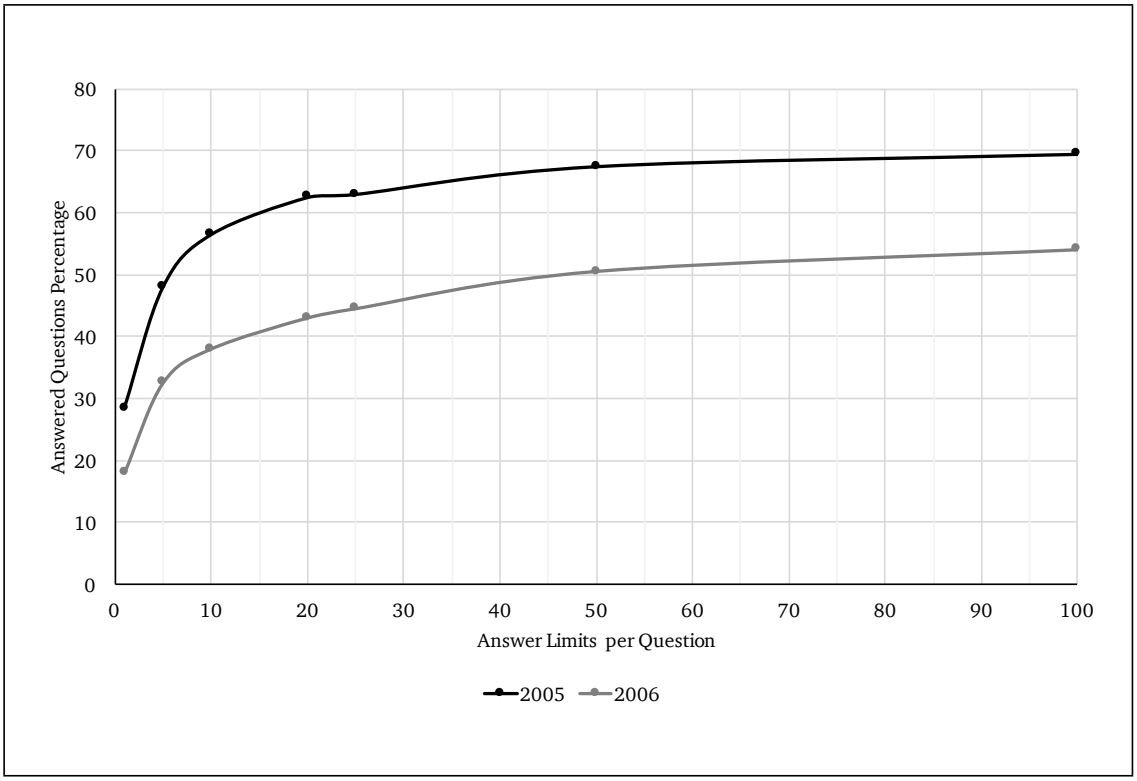

Figure 6.2: Retrieved answers with different limits for QA@CLEF 2005 and 2006

question for both years, RAPPORT was able to answer correctly to 24.75% of the factoid questions, 17.05% of the definition questions, and 33.33% of list questions. As for the results by question type, RAPPORT answered correctly to 28.33% of location, 17.95% of measure, 14.29% of object, 23.19% of organization, 9.46% of other, 24.14% of person, and 40.00% of time questions.

Doing the same analysis just for the year 2006, for the limit of 10 answers per question, the system was able to answer correctly to 33.79% of the factoid questions, 45.65% of the definition questions, and 44.44% of list questions. As for the results by question type, the system answered correctly to 28.00% of location, 38.10% of measure, 57.14% of object, 41.94% of organization, 32.08% of other, 41.86% of person, and 50.00% of time questions.

Finally, the system was able to present no answer to 40% of the questions whose expected answer was nil.

6.2.3 Evaluation using both CHAVE and Wikipedia

Here we focus on the years of 2007 and 2008, where the Portuguese Wikipedia was used together with CHAVE. In these two years, linked questions were introduced — questions that, to be correctly interpreted, depend on others or their answers. The approach follows the same structure from when it as used only with CHAVE, with a few

changes to deal with Wikipedia articles.

Officially, of the 400 questions used in 2007 and 2008, 99 had their answers found on CHAVE, 276 had their answers found on Wikipedia articles, and 25 were nil. However, perusing to the official results, we have found that some of the questions had answers found in both corpora.

In part due to this situation, but mainly because the two corpora are processed in slightly different ways, and the documents themselves have different structures and attributes, we have decided to have independent document indices for CHAVE and for Wikipedia. The fact and the sentence indices are shared by both corpora, as they are structurally identical.

For those two years, using both CHAVE and Wikipedia, we were able to get the right answer to 13.00% of the questions, with a limit of one answer per question. If that limit is changed to 10, we were able to answer 29.00% of the questions. Tables 6.5 and 6.6 show the results for both the 2007 and 2008 years, respectively.

Table 6.5: Comparison of the results at QA@CLEF 2007

Approach	Accuracy for multiple answer limits per question (%)						
	1	**5**	**10**	**20**	**25**	**50**	**100**
Esfinge	8.00	—	—	—	—	—	—
Senso	42.00	—	—	—	—	—	—
Priberam's	50.50	—	—	—	—	—	—
RAPOSA	20.00	—	—	—	—	—	—
QA@L^{2}F	13.00	—	—	—	—	—	—
RAPPORT	9.50	18.50	26.00	30.50	32.00	38.50	40.50

Table 6.6: Comparison of the results at QA@CLEF 2008

Approach	Accuracy for multiple answer limits per question (%)						
	1	**5**	**10**	**20**	**25**	**50**	**100**
Esfinge	23.50	—	—	—	—	—	—
Senso	46.50	—	—	—	—	—	—
Priberam's	63.50	—	—	—	—	—	—
RAPOSA	14.50	—	—	—	—	—	—
QA@L^{2}F	20.00	—	—	—	—	—	—
IdSay	32.50	—	—	—	—	—	—
RAPPORT	16.50	29.00	32.00	38.00	38.50	44.00	47.00

On the answers that have not been found, we have determined that in some cases the fault is due to questions depending on information contained in other questions or their answers. In both years, 51 of the 200 questions were *linked questions*.

If just the *unlinked* (or *first*) questions are considered, the results improve, with a combined accuracy for both years of 17.45%, 31.88%, 38.93%, 45.97%, 47.32%, 55.37%, and 59.73% for each of the cut-off points being used in 2007 (1, 5, 10, 20, 25, 50 and 100).

A graphic representation of the results of our system for different cut-off points in those two years is presented in Fig. 6.3.

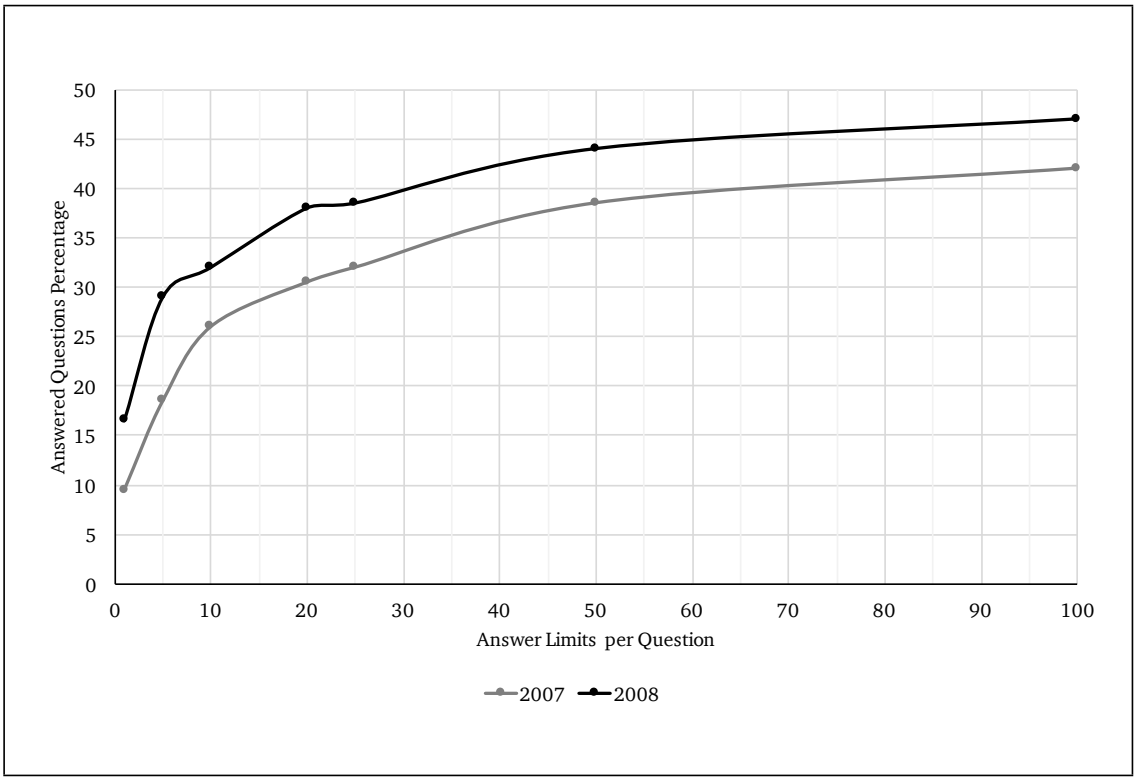

Figure 6.3: Retrieved answers with different limits for QA@CLEF 2007 and 2008

Finally, regarding the categories of the questions, for a limit of one answer per question, for 2007, RAPPORT was able to answer correctly to 13.96% of the factoid questions, 15.52% of the definition questions, and 0.0% of list questions. As for the results by question type, RAPPORT answered correctly to 15.00% of count, 15.07% of location, 16.13% of measure, 16.67% of object, 13.64% of organization, 1.27% of other, 22.53% of person, and 18.18% of time questions.

As for the nil questions, although in Giampiccolo et al. (2008) is mentioned that there were such questions also in 2007, in the files we got with the questions and answers of QA@CLEF, there we no questions with NIL as the answer or even with missing answers. For 2008, those questions were easily identified, but the system failed to provide no answers for them.

6.2.4 Overall Analysis of the Results

In Table 6.7 is presented a general compilation of the results for all years from 2004 to 2008, with the limit of one answer per question. Our system is competitive for the years 2004 to 2006, when just CHAVE was used (with *Público* alone in 2004), achieving

the best result in 2005, only behind Priberam's. In 2007 and 2008, however, it has the lowest scores.

Overall, RAPPORT performs better with CHAVE than with Wikipedia. The main issues are arguably related to these issues: CHAVE documents are typically smaller, more assertive, more homogeneous in the writing style, and better structured, and the sentences are more probable to encapsulate in themselves all the answer (although there is still an evident need coreference resolution); Wikipedia documents are more heterogeneous in the writing style, the sizes vary greatly, from small articles to really large ones, and their structure is less formal.

As for the size of the corpus, at the time the snapshot was created, in 2006, Wikipedia had next to 200,000 articles, while CHAVE had around 150,000 news articles. Although there is a non-negligible difference in size, the reason for such difference on the performance of the system must lie in the documents of both corpora and how they are processed, and on the quality and quantity of the facts extracted.

Table 6.7: Comparison of the results at QA@CLEF 2004 to 2008

Approach	Overall Accuracy (%)					
	2004	**2005**	**2006**	**2006|10**	**2007**	**2008**
Esfinge	15.08	23.00	24.50	23.56	8.00	23.50
Senso	28.54	25.00	—	—	42.00	46.50
Priberam	—	64.50	67.00	67.00	50.50	63.50
NILC	—	—	1.50	1.50	—	—
RAPOSA	—	—	13.00	18.83	20.00	14.50
QA@L^{2}F	—	—	—	—	13.00	20.00
IdSay	—	—	—	—	—	32.50
RAPPORT	17.59	28.50	18.00	38.00	9.50	16.50

Apart the differences between the two corpora, there are other issues that the system has address in the future: NER has to be better — as good as the OpenNLP's NER tool is, we do need better results —, and the same applies at least to the chunker and the dependency parser, responsible for the creation of the chunks used.

Besides that, there are certainly also shortcomings in the creation of the facts, mainly on the phrase chunks that are close together, as opposed to the dependency chunks, that should and must be addressed, in order to improve and create more facts. Furthermore, there are questions that refer to entities that fail to be identified as such by our system, and so no facts were created for them when processing the sentences.

6.2.5 A Study on Diverse Parameter Configurations

We produce here a study on the values of two parameters and discuss how they affect the results, mostly at the query building level and at the answer processing level. Fact extraction and storage are so crucial to the approach, with fact extraction resulting

from a long line of minimal improvements, that trying different parameters on those stages always affect the outcome of the system. We have used as reference a limit of one answer per question.

Changing the Answer Limits per Question

In Fig. 6.4, and also in Table 6.8, we can see the overall variation in the number of retrieved answers in response to different answer limits, from 2004 to 2008 — in essence, a superposition of the previous graphics regarding cut-off points. We can observe that from all the correct answers the system can retrieve, half of them are in the first five answers, and more than three quarters are under the 25 answers per question limit.

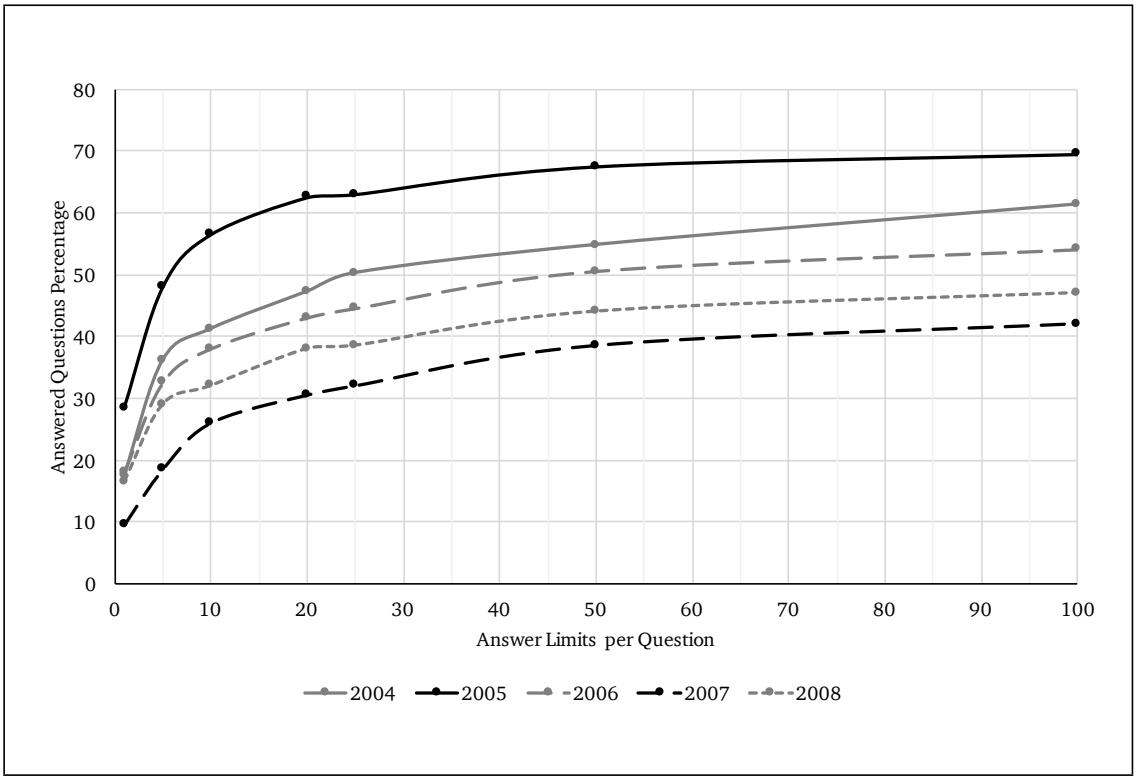

Figure 6.4: Retrieved answers with different limits for QA@CLEF 2004 to 2008

An issue with RAPPORT is that, depending on the years, it can not retrieve the right answers to between 30% and 50% of the questions, even for a limit of 100 answers per question, and looking at the curve in the graphic, we can assume that those values are stable no matter the limit. Typically, those questions are either too hard, with their answers scattered among multiple sentences in a text, or too difficult for the system to interpret. Other issues include: questions that depend on document metadata, that the system does not address currently; questions that do not revolve around named entities; or simpler cases, like the failure to recognize entities in sentences, and to produce the expected facts, or even questions that do not revolve around entities.

Table 6.8: Retrieved answers with different limits for QA@CLEF 2004 to 2008

Years / Limits	1	5	10	20	25	50	100
2004	17.59	36.18	41.21	47.24	50.25	54.77	61.31
2005	28.50	48.00	56.50	62.50	63.00	67.50	69.50
2006	18.00	32.50	38.00	43.00	44.50	50.50	54.00
2007	9.50	18.50	26.00	30.50	32.00	38.50	42.00
2008	16.50	29.00	32.00	38.00	38.50	44.00	47.00

With and Without Query Expansion

For testing queries both with and without expansion of the terms in the query, two types of queries where tested: one with just the keywords retrieved from the topic question, and another expanding verbs and nouns with synonyms, and also replacing toponyms by demonyms and *vice-versa*. As the best results came marginally without query expansion, Table 6.9 presents the results with query expansion, for purposes of comparison.

Table 6.9: Results with query expansion

Years / Limits	1	5	10	20	25	50	100
2004	14.57	31.66	36.18	42.21	45.23	50.25	58.29
2005	22.50	46.00	55.00	61.00	63.00	67.50	69.50
2006	16.50	31.50	38.50	43.00	44.00	48.50	50.50
2007	8.50	17.00	25.00	29.50	31.00	38.00	41.00
2008	17.50	29.00	31.00	37.00	37.50	45.50	47.50

Against our initial expectation, the use of query expansion (with synonym expansion, toponym and demonym expansion, and nominalization of verbs) does not provide noticeable gains in terms of results — it gets better results just in two occasions, for a limit of one answer. For other limits, the results except for a few cases, are also globally worse, even if only marginally.

We believe that the reason for such results is that creating wider queries also introduces noise in the scoring of the potential answers. Additionally, most of questions use the same words that are expected in the answers, as a result of the evaluators working closely with the texts for the creation of the questions.

Answer Ordering Based on Presence in Facts

An alternative we considered for ordering the final answers, was using the number of facts that contained the answers in them (in the subject or the object). The rationale was a more frequent answer would be preferable to a rarer one. However, as it happened in the previous experience, the results are lower, even if also marginally. For a limit of one answer, it does have better results in two years, and a few more in other limits, but overall the results are worse than the order of the answers directly based on the

facts and corresponding sentences hit score. The results for this study are present in Table 6.10.

Table 6.10: Results with ordering based on fact frequency

Years / Limits	1	5	10	20	25	50	100
2004	22.61	36.68	40.70	43.72	46.73	52.76	61.31
2005	27.00	44.00	55.00	60.00	62.50	65.50	69.50
2006	18.50	30.00	36.50	44.00	46.00	52.00	53.50
2007	6.00	15.00	23.00	27.50	29.50	35.00	40.50
2008	15.50	25.00	30.50	37.50	38.00	45.00	48.00

Chapter 7

Conclusions

In this chapter we present concluding remarks of the work done in the scope of this thesis. We discuss what are the contributions this system brings of new to the field, what works and what does not, providing explanations for both. We also refer to future paths available for exploration and improvement of the system.

7.1 Final Discussion

In the work described in this thesis, we have developed and implemented a new approach to question answering for the Portuguese language, based on the use of facts both for storing information and for presenting answers as short passages. For that to be possible, we have:

- studied and analyzed the mechanisms and approaches for question answering and related work;
- surveyed which NLP tools existed and, of those, which can be used or adapted — determining, at the same time, the tools that would have to be developed for the task at hands;
- analyzed sentence structure and contents in order to extract relevant information that can be used later to answer questions, including the extraction of short facts about or related to named entities;
- studied how facts can be used for presenting answers as short passages (opposing to full sentences or documents);
- focused on the Portuguese language, while checking what has been done in other (related) languages.

Retrospectively, we have to concede that the plan was overly ambitious, mainly due to the fact that question answering is tightly dependent on many NLP tasks and on each of them. Some of those tasks are difficult to tackle, with limited or even non-existing tools for Portuguese in a ready to use basis. On top of that, there are actually two big problems QA systems have to address: one problem is to get the candidate answers — the shorter the answer, the harder the task; the other problem is to rank the candidate answers, in order to get the most useful on top of the others. That is, QA is, without question, a hard problem to address.

Nevertheless, for developing a new QA system, the most common and known question answering approaches were studied and, when possible, deconstructed, specially focusing on those targeting the Portuguese language, identifying the key tasks: sentence splitting, tokenization, POS tagging, named entity recognition, phrase chunking, storage, querying, etc. Each of these tasks was eventually reproduced or adapted, and applied in order for us to have a working system, and then evolved, or modified. Some of the tools that support those tasks were created from scratch, based on the study of other approaches.

In the context of the QA@CLEF tracks, the QA system should be able to return to the user just the strict answers, in the form of short passages, in a timely manner. Bearing that in mind, we have chosen to store in the indices just documents and sentences that would have associated facts, leaving out sentences (and possibly documents) that would not have any facts extracted from them. This has, however, at least one adverse consequence: documents or sentences that the system fails to identify as having facts on them are eventually discarded. The reason for such behavior is that currently there are only facts regarding named entities (or alternatively proper nouns). We have made this choice based on the fact that majority of the questions are about entities, and also in order keep the number of extracted facts manageable, avoiding at the same time irrelevant ones — this however must the addressed and eventually changed in a future iteration of the system. As such, if a sentence has no named entities in it, and therefore no associated facts, it is ignored.

Facts can also be extracted from sentences without named entities or proper nouns, using just the structure of the sentence, but we have chosen to leave that for a future iteration of the system, as it would increase manyfold the number of extracted facts, with the associated costs in terms of storage, search and ranking of the facts as the source for answers.

The use of facts also eases the retrieval of answer passages, as facts can be considered relations between small snippets of text, ending up being used as the source of the answer passages. As a matter of fact, for all it may be worth, the 2008 edition of the NLP reference book *Speech and Language Processing* (Jurafsky and Martin, 2008), does

not yet mention the use of open information extraction applied to QA. The use of facts is something that has emerged in the last years, contemporaneously to the work here presented and described.

The final results show that improvements were possible, mainly by pre-processing the corpus and selecting key elements of information under facts (triples), that would be later used to build the answers to user questions, using the contents of those facts as short answers or passages.

Although we were not able to set new reference values for the benchmarks used, namely for the QA@CLEF Portuguese tracks, we are confident that the use of facts for extracting, storing and retrieving information provides a new and valuable approach for QA for Portuguese. It can be a stepping stone for further improvements, as even in this stage, it gets close to, or surpasses, the best academic approaches, depending on the year editions of the benchmark considered. A continued investment, with more resources, on the proposed path may be able to achieve better results. Additionally, one can consider the extraction of facts as a way of summarizing texts, applying the work done to other fields related to question answering. Overall, and despite any shortcomings, the system performs well at providing the expected short answers.

7.2 Contributions

The work described in this thesis has resulted in several contributions, both in the specific field of QA and in the broader scope of natural language processing.

On the question answering field, we have developed a new approach for Portuguese based on the use of facts, on pair with other academic systems for the benchmark used. The approach is also adaptable and easily applicable to other corpora, being an open-world approach.

On the general field of natural language processing, we have developed a suite of tools for addressing the many tasks that support a QA system. Of those tools, the most prominent one is the LEMPORT lemmatizer, developed from scratch, which has been published and made publicly available. Other tools also developed from the ground up are the fact extractor (FACPORT), an essential tool in our QA system, the nominalizer (NOMPORT), and the toponimizer (TOPPORT).

That suite also includes tools based on the *Apache OpenNLP* suite and on *MaltParser*. In such case, tools have been adapted for addressing minor issues and for improving their output, had models specially trained, or just have been adapted for a better integration in our system. For instance, the tokenizer had its input pre-processed in order to group or split words (e.g., proper nouns or verbal inflections), improving its output for further use in the part-of-speech tagger and other tools.

The suite of tools developed, which have been made available to other elements of our research group, free for download from GitHub, and for public use, comprehend the following:

- CHKPORT, the chunker: `http://github.com/rikarudo/ChkPORT`;
- DEPPORT, the dependency parser: `http://github.com/rikarudo/DepPORT`;
- ENTPORT, the NER tool: `http://github.com/rikarudo/EntPORT`;
- FACPORT, the fact extractor tool: `http://github.com/rikarudo/FacPORT`;
- LEMPORT, the lemmatizer: `http://github.com/rikarudo/LemPORT`;
- NOMPORT, the nominalizer: `http://github.com/rikarudo/NomPORT`;
- SENPORT, the sentence splitter: `http://github.com/rikarudo/SenPORT`;
- TAGPORT, the POS tagger: `http://github.com/rikarudo/TagPORT`;
- TOKPORT, the tokenizer: `http://github.com/rikarudo/TokPORT`;
- TOPPORT, the toponimizer: `http://github.com/rikarudo/TopPORT`.

Together with the tools, we will also be granting full access in the near future to the RAPPORT QA system in `http://github.com/rikarudo/RAPPORT` for whoever wants to test it or improve it.

In addition to the tools, we have also contributed with scientific publications, in an international journal, and presented in international events, by ourselves and in collaboration with others:

- Participation in the information retrieval task PÁGICO:
  - Ricardo Rodrigues, Hugo Gonçalo Oliveira, and Paulo Gomes. Uma Abordagem ao Págico baseada no Processamento e Análise de Sintagmas dos Tópicos. *LinguaMÁTICA*, 4(1):31–39, April 2012
- Creation of the LEMPORT lemmatizer:
  - Ricardo Rodrigues, Hugo Gonçalo Oliveira, and Paulo Gomes. LemPORT: a High-Accuracy Cross-Platform Lemmatizer for Portuguese. In Maria João Varanda Pereira, José Paulo Leal, and Alberto Simões, editors, *Proceedings of the 3rd Symposium on Languages, Applications and Technologies (SLATE '14)*, OpenAccess Series in Informatics, pages 267–274, Germany, June 2014. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Dagstuhl Publishing

- Development of the RAPPORT QA System:
  - Ricardo Rodrigues and Paulo Gomes. RAPPORT — A Portuguese Question-Answering System. In Francisco Câmara Pereira, Penousal Machado, Ernesto Costa, and Amílcar Cardoso, editors, *Progress in Artificial Intelligence — 17th Portuguese Conference on Artificial Intelligence (EPIA 2015)*, volume 9273 of Lecture Notes in Computer Science, pages 771–782, Coimbra, Portugal, September 2015. Springer
  - Ricardo Rodrigues and Paulo Gomes. Improving Question-Answering for Portuguese using Triples Extracted from Corpora. In *Proceedings of the 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of LNCS/LNAI, pages 25–37, Tomar, Portugal, July 2016. Springer
- Other collaborations:
  - Daniel Diéguez, Ricardo Rodrigues, and Paulo Gomes. Using CBR for Portuguese Question Generation. In *Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)*, pages 328–341, Lisbon, Portugal, October 2011. APPIA
  - Ana Oliveira Alves, Ricardo Rodrigues, and Hugo Gonçalo Oliveira. ASAPP: Alinhamento Semântico Automático de Palavras Aplicado ao Português. *LinguaMÁTICA*, 8(2):43–58, December 2016
  - Hugo Gonçalo Oliveira, Ana Oliveira Alves, and Ricardo Rodrigues. Gradually Improving the Computation of Semantic Textual Similarity in Portuguese. In Eugénio Oliveira, João Gama, Zita Vale, and Henrique Lopes Cardoso, editors, Progress in Artificial Intelligence — 18th Portuguese Conference on Artificial Intelligence (EPIA 2017), volume 10423 of Lecture Notes in Computer Science, pages 841–854, Porto, Portugal, September 2017. Springer

7.3 Future Work

In our opinion, important steps were given on automatic question answer for Portuguese. There is, as always, room for improvement. Not surprisingly, much of what can be considered for future improvements is scattered over the four modules that compose RAPPORT, given the multiple tasks found in each of them. For instance, the fact extraction task must be improved. First, facts should not be restricted only to named entities, or alternatively proper nouns; second, there should be filters for removing facts possessing no relevant or too generic information, based on their contents.

It is also our intention to test, in addition to synonyms, the use of hyponyms, hypernyms, and other relations between tokens or lemmas, in order to expand the queries made to the indices, which will increase the number of retrieved sentences and facts, using a Wordnet-like resource, such as Onto.PT (Gonçalo Oliveira, 2013). That, on the other end, will impose a deeper analysis of how to filter the *right* facts from those that are not relevant.

We are currently studying a way of relating words, namely verbs and nouns, such as "*ensinar*" (*to teach*) and "*professor*" (*teacher* or *professor*), improving the *nominalizer*. In nouns that are directly related to verbs, a small set of rules is enough, but in this example some sort of list has to be compiled.

Another aspect that should be considered is the use of coreference resolution (Jurafsky and Martin, 2008) in order to improve the recall of triples by way of replacing, for instance, pronouns with the corresponding, if any, named entities, and hence increasing the number of usable facts.

We must also revise all the external tools used in RAPPORT, looking for ways to improve them. For instance, better results from the NER tool would affect the output of many tasks, including fact extraction.

Finally, we should also address questions that depend on metadata of documents, such as the dates in news articled, that can act as a timestamp for data and information.

We believe that expanding the queries using the above techniques, and creating better rules or models for extracting triples can achieve better results in a short or moderate time span, bearing in mind also that ranking also has to improve.

Even what can be considered more stable tools, such as LEMPORT, can be improved. As stated previously at the end of Subsection 5.2.5, the lemmatizer can benefit from improvements regarding, for instance, how hyphenated words, multiword expressions, and oblique cases are processed.

7.4 Concluding Remarks

We would like to end up by stating that, although conscious of its limitations, we believe RAPPORT is a substantive contribution to natural language processing targeting the Portuguese language.

Question answering is dependent on many individual tasks, on the one hand befitting from them, but also on the other hand amplifying each of their faults. Any fault in the first tasks performed for processing the text is propagated to the subsequent tasks. Ideally, question answering should just focus on getting candidate answers (including extracting or retrieving them) and ranking them. However, question answering is eventually built on top of a chain of natural language processing tasks.

Out of *naïveté*, when we first delved into question answering for Portuguese, we have just focused on the tasks directly related to the final stages of a QA system, assuming that the tools for all the tasks that would precede them were easily available and mature. They were neither. Even today we are in need of an anaphora resolution tool for Portuguese that we can incorporate in our system (although a few of such systems exist, as it is the case of LinkPeople (Garcia and Gamallo, 2014), or COPR (Fonseca et al., 2016), which is however only accessible as an web demonstration, at the time of the writing of this thesis).

Then, there is the set of tools available in *Apache OpenNLP*, of which only three have pre-trained models for Portuguese (the sentence splitter, the tokenizer, and the POS tagger). We were able to train models for the chunker and the entity finder. But, in both cases, though helpful, we are *limited* by them, even after pre- and post-processing are applied to the tools. The same applies to *MaltParser*. That is to say, even if we continue to use those tools, we have to pay more attention to the models, and to the pre- and post-processing of both their input and output.

As for the lemmatizer, having not found any tool that can be easily integrated in our system, we have build LEMPORT, which ended being one of the best (if not the best currently) for Portuguese.

Aware of all these difficulties, we have opted for making freely available all the tools we have developed, tweaked or adapted, for easing the efforts of those that will come after us, hoping at the same time to benefit from the same behavior from others. Only then can question answering for Portuguese, and other NLP related applications and tasks, evolve and get on par with the best systems for other languages.

It takes a village!

References

Abney, S. P. Part-of-Speech Tagging and Partial Parsing. In *Corpus-Based Methods in Language and Speech Processing*, pages 118–136. Springer Netherlands, Dordrecht, Netherlands, 1997.

Afonso, S., Bick, E., Haber, R., and Santos, D. "Floresta Sintá(c)tica": a Treebank for Portuguese. In Rodríguez, M. G. and Araujo, C. P. S., editors, *Proceedings of the 3rd Language Resources and Evaluation Conference (LREC 2002)*, pages 1698–1703, Paris, 2002. ELRA.

Akbik, A. and Löser, A. KrakeN: N-ary Facts in Open Information Extraction. In *Proceedings of the Joint Workshop on Automatic Knowledge Base Construction and Web-scale Knowledge Extraction (AKBC-WEKEX 2012)*, pages 52–56, Montreal, Canada, June 2012.

Alfonseca, E. and Manandhar, S. An Unsupervised Method for General Named Entity Recognition and Automated Concept Discovery. In *Proceedings of the 1st International Conference on General WordNet*, pages 34–43, Mysore, India, 2002.

Amaral, C., Figueira, H., Martins, A., Mendes, A., Mendes, P., and Pinto, C. Priberam's Question Answering System for Portuguese. In *Proceedings of the 6th Workshop of the Cross-Language Evaluation Forum (CLEF 2005)*, pages 410–419, Berlin Heidelberg, 2005. Springer-Verlag.

Amsler, R. A. A Taxonomy for English Nouns and Verbs. In *Proceedings of the 19th Annual Meeting of the Association for Computational Linguistics*, pages 133–138, 1981.

Baeza-Yates, R. and Ribeiro-Neto, B. *Modern Information Retrieval*. ACM Press, New York, USA, 1999.

Balage Filho, P. P., de Uzêda, V. R., Pardo, T. A. S., and das Graças Volpe Nunes, M. Using a Text Summarization System for Monolingual Question Answering. In *CLEF 2006 Working Notes*, 2006.

Balage Filho, P. P., Pardo, T. A. S., and das Graças Volpe Nunes, M. Summarizing Scientific Texts: Experiments with Extractive Summarizers. In *Proceedings of the 7th International Conference on Intelligent Systems Design and Applications*, pages 520–524. IEEE Computer Society, 2007.

Banko, M. and Etzioni, O. The Tradeoffs Between Open and Traditional Relation Extraction. In *Proceedings of the 46th Annual Meeting of the Association for Computational Linguistics (ACL-08)*, pages 28–36, 2008.

Banko, M. and Moore, R. C. Part of Speech Tagging in Context. In *Proceedings of the 20th International Conference on Computational Linguistics (COLING '04)*, pages 556–561, Stroudsburg, PA, USA, 2004.

Bast, H. and Haussmann, E. Open Information Extraction via Contextual Sentence Decomposition. In *Proceedings of the 7th International Conference on Semantic Computing (ICSC '13)*, pages 154–159, Washington, DC, USA, 2013. IEEE Computer Society.

Bick, E. PALAVRAS, a Constraint Grammar-based Parsing System for Portuguese. In Sardinha, T. B. and de Lurdes São Bento Ferreira, T., editors, *Working with Portuguese Corpora*, chapter 14, pages 279–302. Bloomsburry Academic, London, UK, 2014.

Bizer, C., Lehmann, J., Kobilarov, G., Auer, S., Becker, C., Cyganiak, R., and Hellmann, S. DBpedia — A Crystallization Point for the Web of Data. *Web Semantics: Science, Services and Agents on the World Wide Web*, 7(3):154–165, September 2009.

Bos, J. and Markert, K. Combining Shallow and Deep NLP Methods for Recognizing Textual Entailment. In *Proceedings of the Pascal Challenges Workshop on Recognising Textual Entailment*, Southhampton, UK, April 2005.

Branco, A. and Silva, J. A Suite of Shallow Processing Tools for Portuguese: LX-Suite. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics (EACL '06): Posters & Demonstrations*, pages 179–182, Trento, Italy, 2006.

Branco, A. and Silva, J. Very High Accuracy Rule-Based Nominal Lemmatization with a Minimal Lexicon. In *Actas do XXII Encontro Nacional da Associação Portuguesa de Linguística*, pages 169–181, Lisbon, Portugal, 2007.

Branco, A., Rodrigues, L., Silva, J., and Silveira, S. Real-Time Open-Domain QA on the Portuguese Web. In *Proceedings of the 11th Ibero-American Conference on AI: Advances in Artificial Intelligence*, Berlin Heidelberg, 2008a. Springer-Verlag.

Branco, A., Rodrigues, L., Silva, J., and Silveira, S. XisQuê: An Online QA Service for Portuguese. In *Proceedings of the 8th International Conference on Computational Processing of the Portuguese Language (PROPOR '08)*, pages 322–331, Berlin Heidelberg, 2008b. Springer-Verlag.

Brill, E. A Simple Rule-Based Part of Speech Tagger. In *Proceedings of the 3rd Conference on Applied Natural Language Processing*, pages 152–155, Morristown, New Jersey, USA, 1992. Association for Computational Linguistics.

Brill, E., Dumais, S., and Banko, M. An Analysis of the AskMSR Question-Answering System. In *Proceedings of the ACL-02 Conference on Empirical Methods in Natural Language Processing*, volume 10, pages 257–264, Morristown, New Jersey, USA, 2002. Association for Computational Linguistics.

Brin, S. and Page, L. The Anatomy of a Large-Scale Hypertextual Web Search Engine. In *Computer Networks and ISDN Systems*, volume 30, pages 107–117, 1998.

Buchholz, S. and Marsi, E. CoNLL-X shared task on Multilingual Dependency Parsing. In *Proceedings of the 10th Conference on Computational Natural Language Learning (CoNLL-X)*, pages 149–164, New York, USA, June 2006.

Cardoso, N. Medindo o Princípio Semântico. *LinguaMÁTICA*, 4(1):41–48, April 2012a.

Cardoso, N. Rembrandt — A Named-Entity Recognition Framework. In *Proceedings of the 8th Language Resources and Evaluation Conference (LREC 2012)*, pages 1240–1243, Istanbul, Turkey, 2012b.

Cardoso, N., Cruz, D., Chaves, M., and Silva, M. J. Using Geographic Signatures as Query and Document Scopes in Geographic IR. In *Proceedings of the Workshop of the Cross-Language Evaluation Forum for European Languages (CLEF 2007)*, pages 802–810, 2008.

Carreras, X., Chao, I., Padró, L., and Padró, M. FreeLing: An Open-Source Suite of Language Analyzers. In *Proceedings of the 4th International Conference on Language Resources and Evaluation (LREC 2004)*, pages 239–242, 2004.

Carvalho, G., de Matos, D. M., and Rocio, V. IdSay: Question Answering for Portuguese. In Peters, C., Deselaers, T., Ferro, N., Gonzalo, J., Jones, G. J., Kurimo, M., Mandl, T., Peñas, A., and Petras, V., editors, *Evaluating Systems for Multilingual and Multimodal Information Access: 9th Workshop of the Cross-Language Evaluation Forum (CLEF 2008)*, volume 5706 of *LNCS*, pages 345–352, Berlin Heidelberg, September 2009. Springer-Verlag.

Carvalho, G., de Matos, D. M., and Rocio, V. Robust Question Answering. In *PhD and MSc/MA Dissertation Contest of the 10th International Conference on Computational Processing of the Portuguese Language (PROPOR 2012)*, Coimbra, Portugal, April 2012.

Coheur, L., Mendes, A., Guimarães, J., Mamede, N. J., and Ribeiro, R. QA@L^{2}F, Second steps at QA@CLEF. In *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark, September 2008.

Costa, D., Gonçalo Oliveira, H., and Pinto, A. "In reality there are as many religions as there are papers" — First Steps Towards the Generation of Internet Memes. In *Proceedings of the 6th International Conference on Computational Creativity*, pages 300–307, 2015.

Costa, L. F. Esfinge — A Modular Question Answering System for Portuguese. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2006 Workshop (CLEF 2006)*, Alicante, Espanha, September 2006a.

Costa, L. F. 20th Century Esfinge (Sphinx) Solving the Riddles at CLEF 2005. In *Accessing Multilingual Information Repositories: 6th Workshop of the Cross-Language Evalution Forum (CLEF 2005) — Revised Selected Papers*, pages 467–476, Berlin Heidelberg, 2006b. Springer-Verlag.

Costa, L. F. Esfinge — A Question Answering System in the Web using the Web. In *Proceedings of the Demonstration Session of the 11th Conference of the European Chapter of the Association for Computational Linguistics*, pages 410–419, Trento, Italy, April 2006c. Association for Computational Linguistics.

Costa, L. F. Esfinge at CLEF 2008: Experimenting with Answer Retrieval Patterns. Can They Help? In *Cross Language Evaluation Forum: Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark, September 2008.

Covington, M. A. A Fundamental Algorithm for Dependency Parsing. In *Proceedings of the 39th Annual ACM Southeast Conference*, pages 95–102, Athens, Georgia, USA, 2001.

Cunha, C. and Cintra, L. *Nova Gramática do Português Moderno*. Edições João Sá da Costa, Lisbon, Portugal, 17th edition, 2002.

da Silva, B. C. D. and de Moraes, H. R. A Construção de um Thesaurus Electrônico para o Português do Brasil. *Alfa — Revista de Linguística*, 47(2), 2003.

del Corro, L. and Gemulla, R. ClausIE: Clause-Based Open Information Extraction. In *Proceedings of the 22nd World Wide Web Conference (WWW-2013)*, pages 355–366, Rio de Janeiro, Brazil, May 2013.

Diéguez, D., Rodrigues, R., and Gomes, P. Using CBR for Portuguese Question Generation. In *Proceedings of the 15th Portuguese Conference on Artificial Intelligence (EPIA 2011)*, pages 328–341, Lisbon, Portugal, October 2011. APPIA.

Ding, L., Kolari, P., Ding, Z., Avancha, S., Finin, T., and Joshi, A. Using Ontologies in the Semantic Web: A Survey. Technical Report CS-05-07, University of Maryland Baltimore County (Department of Computer Science and Electrical Engineering), Baltimore, Maryland, USA, July 2005.

Drumond, L. and Girardi, R. A Survey of Ontology Learning Procedures. In *Proceedings of the 3rd Workshop on Ontologies and Their Applications (WONTO 2008)*. Association for Computational Linguistics, 2008.

Edmunds, A. and Morris, A. The Problem of Information Overload in Business Organisations: a Review of the Literature. *International Journal of Information Management*, 20:17–28, 2000.

Eleutério, S., Ranchhod, E. M., Mota, C., and Carvalho, P. Dicionários Electrónicos do Português. Características e Aplicações. In *Actas del VIII Simposio Internacional de Comunicación Social*, pages 637–643, 2003.

Estival, D., Nowak, C., and Zschorn, A. Towards Ontology-Based Natural Language Processing. In *Proceedings of the 4th Workshop on NLP and XML (NLPXML-2004)*, Barcelona, Spain, 2004. Association for Computational Linguistics.

Etzioni, O., Fader, A., Christensen, J., Soderland, S., and Mausam. Open Information Extraction: the Second Generation. In *Proceedings of the 22nd International Joint Conference on Artificial Intelligence (IJCAI '11)*, volume I, pages 3–10, Barcelona, Catalonia, Spain, 2011.

Fader, A., Soderland, S., and Etzioni, O. Identifying Relations for Open Information Extraction. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing (EMNLP '11)*, pages 1535–1545, Edinburgh, Scotland, UK, July 2011.

Feigenbaum, E. A. Some Challenges and Grand Challenges for Computational Intelligence. *Journal of the ACM*, 50(1):32–40, January 2003.

Feldman, S. and Sherman, C. The High Cost of Not Finding Information. White Paper, IDC, Framingham, MA, USA, July 2001.

Fellbaum, C. *WordNet: An Electronic Lexical Database (Language, Speech, and Communication)*. A Bradford Book, May 1998.

Ferreira, L., Teixeira, A., and da Silva Cunha, J. P. REMMA — Reconhecimento de Entidades Mencionadas do MedAlert. In Mota, C. and Santos, D., editors, *Desafios na Avaliação Conjunta do Reconhecimento de Entidades Mencionadas: O Segundo HAREM*, chapter 12, pages 213–229. Linguateca, 2008.

Flores, F. N., Moreira, V. P., and Heuser, C. A. Assessing the Impact of Stemming Accuracy on Information Retrieval. In Pardo, T. A. S., Branco, A., Klautau, A., Vieira, R., and de Lima, V. L. S., editors, *Proceedings of the 9th International Conference on Computational Processing of the Portuguese Language (PROPOR '10)*, volume 6001 of *Lecture Notes in Computer Science*, pages 11–20, Berlin, Heidelberg, 2010. Springer-Verlag.

Fonseca, E., Vieira, R., and Vanin, A. CORP: Coreference Resolution for Portuguese. In *Proceedings of the Demonstration Session of the 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, pages 9–11, Tomar, Portugal, 2016.

Forner, P., Peñas, A., Agirre, E., Alegria, I., Forascu, C., Moreau, N., Osenova, P., Prokopidis, P., Rocha, P., Sacaleanu, B., Sutcliffe, R., and Sang, E. T. K. Overview of the CLEF 2008 Multilingual Question Answering Track. In *Evaluating Systems for Multilingual and Multimodal Information Access*, volume 5706 of *LCNS*, pages 262–295, Berlin Heidelberg, 2009. Springer-Verlag.

Freitas, C., Rocha, P., and Bick, E. Floresta Sintá(c)tica: Bigger, Thicker and Easier. In *Proceedings of the 8th International Conference on Computational Processing of the Portuguese Language (PROPOR '08)*, pages 216–219. Springer-Verlag, 2008.

Gamallo, P. An Overview of Open Information Extraction. In Pereira, M. J. V., Leal, J. P., and Simões, A., editors, *Proceedings of the 3rd Symposium on Languages, Applications and Technologies (SLATE '14)*, OpenAccess Series in Informatics, pages 13–16, Germany, June 2014. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Dagstuhl Publishing.

Gamallo, P., Garcia, M., and Fernández-Lanza, S. Dependency-Based Open Information Extraction. In *Proceedings of the 13th Conference of the European Chapter of the Association for Computational Linguistics (EACL 2012)*, pages 10–18, Avignon, France, April 2012. Association for Computational Linguistics.

Garcia, M. and Gamallo, P. Entity-Centric Coreference Resolution of Person Entities for Open Information Extraction. *Procesamiento del Lenguaje Natural*, 53:25–32, September 2014.

Giampiccolo, D., Forner, P., Herrera, J., Peñas, A., Ayache, C., Forascu, C., Jijkoun, V., Osenova, P., Rocha, P., Sacaleanu, B., and Sutcliffe, R. Overview of the CLEF 2007 Multilingual Question Answering Track. In Peters, C., Jijkoun, V., Mandl, T., Müller, H., Oard, D. W., Peñas, A., Petras, V., and Santos, D., editors, *Advances in Multilingual and Multimodal Information Retrieval*, volume 5152 of *LCNS*, pages 200–236, Berlin Heidelberg, 2008. Springer-Verlag.

Gonçalo Oliveira, H. *Onto.PT: Towards the Automatic Construction of a Lexical Ontology for Portuguese*. PhD thesis, Faculty of Sciences and Technology of the University of Coimbra, 2013.

Gonçalo Oliveira, H. and Gomes, P. Onto.PT: Automatic Construction of a Lexical Ontology for Portuguese. In *Proceedings of the 5th European Starting AI Researcher Symposium (STAIRS 2010)*, Lisbon, Portugal, August 2010. IOS Press.

Gonçalo Oliveira, H., Costa, H., and Gomes, P. Extracção de Conhecimento Léxico-Semântico a partir de Resumos da Wikipédia. In *Proceedings of INForum 2010 — II Simpósio de Informática*, pages 537–548, Braga, Portugal, September 2010.

Gonçalo Oliveira, H., Antón Pérez, L., Costa, H., and Gomes, P. Uma Rede Léxico-Semântica de Grandes Dimensões para o Português, Extraída a partir de Dicionários Electrónicos. *LinguaMÁTICA*, 3(2):23–38, December 2011.

Gonçalo Oliveira, H., Coelho, I., and Gomes, P. Exploiting Portuguese Lexical Knowledge Bases for Answering Open Domain Cloze Questions Automatically. In *Proceedings of the 9th Language Resources and Evaluation Conference (LREC 2014)*, pages 4202–4209, Reykjavik, Iceland, 2014.

Gonçalo Oliveira, H., Costa, D., and Pinto, A. Automatic Generation of Internet Memes from Portuguese News Headlines. In *Proceedings of the 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of *LNAI*, pages 340–346, Tomar, Portugal, July 2016. Springer.

Gonçalo Oliveira, H., Oliveira Alves, A., and Rodrigues, R. Gradually Improving the Computation of Semantic Textual Similarity in Portuguese. In Oliveira, E., Gama, J., Vale, Z., and Cardoso, H. L., editors, *Progress in Artificial Intelligence — 18th Portuguese Conference on Artificial Intelligence (EPIA 2017)*, volume 10423 of *Lecture Notes in Computer Science*, pages 841–854, Porto, Portugal, September 2017. Springer.

Görz, G. and Schellenberger, G. Chunk Parsing in Corpora. In *The Phraseological View of Language: A Tribute to John Sinclair*, pages 269–282. Mouton de Gruyter, November 2011.

Green Jr., B. F., Wolf, A. K., Chomsky, C., and Laughery, K. Baseball: an Automatic Question-Answerer. In *Proceedings of the '61 Western Joint Computer Conference (IRE-AIEE-ACM '61)*, pages 219–224, New York City, New York, USA, May 1961. ACM.

Grefenstette, G. and Tapanainen, P. What is a Word, What is a Sentence? Problems of Tokenization. In *Proceedings of the 3rd International Conference on Computational Lexicography*, pages 79–87, Budapest, Hungary, 1994.

Grishman, R. Information Extraction: Techniques and Challenges. In Pazienza, M. T., editor, *Proceedings of the International Summer School on Information Extraction: A Multidisciplinary Approach to an Emerging Information Technology (SCIE '97)*, pages 10–27, London, UK, 1997. Springer-Verlag.

Gruber, T. R. Toward Principles for the Design of Ontologies Used for Knowledge Sharing. *International Journal Human-Computer Studies*, 43:907–928, 1995.

Hachaj, T. and Ogiela, M. K. Clusters of Trends Detection in Microblogging: Simple Natural Language Processing vs Hashtags — Which is More Informative? In *Proceedings of the 10th International Conference on Complex, Intelligent, and Software Intensive Systems (CISIS)*, pages 119–121. IEEE, July 2016.

Harabagiu, S., Hickl, A., Lehmann, J., and Moldovan, D. Experiments with Interactive Question-Answering. In *Proceedings of the 3rd Annual Meeting on Association for Computational Linguistics*, pages 205–214, Morristown, New Jersey, USA, 2005. Association for Computational Linguistics.

Heeman, P. A. POS Tags and Decision Trees for Language Modeling. In *Proceedings of the Joint SIGDAT Conference on Emperical Methods in Natural Language and Very Large Corpora (EMNLP/VLC-99)*, pages 129–137, College Park, USA, 1999.

Herbelot, A. and Copestake, A. Acquiring Ontological Relationships from Wikipedia Using RMRS. In *Proceedings of the ISWC 2006 Workshop on Web Content Mining with Human Language Technologies*, 2006.

Hermjakob, U. Parsing and Question Classification for Question Answering. In *Proceedings of the ACL-2001 Workshop on Open-Domain Question Answering*, volume 12, pages 1–6, Morristown, New Jersey, USA, 2001. Association for Computational Linguistics.

Hickl, A. and Harabagiu, S. Enhanced Interactive Question-Answering with Conditional Random Fields. In *Proceedings of the Interactive Question Answering Workshop at HLT-NAACL 2006*, volume 12, pages 25–32, New York City, New York, USA, June 2006. Association for Computational Linguistics.

Hickl, A., Wang, P., Lehmann, J., and Harabagiu, S. FERRET: Interactive Question-Answering for Real-World Environments. In *Proceedings of the COLING/ACL 2006 on Interactive Presentation Sessions*, pages 25–28, Morristown, New Jersey, USA, 2006a. Association for Computational Linguistics.

Hickl, A., Williams, J., Bensley, J., Roberts, K., Shi, Y., and Rink, B. Question Answering with LCC's CHAUCER at TREC 2006. In *Proceedings of the 2006 Text Retrieval Conference (TREC 2006)*, Gaithersburg, Maryland, USA, 2006b.

Hickl, A., Roberts, K., Rink, B., Bensley, J., Jungen, T., Shi, Y., and Williams, J. Question Answering with LCC's CHAUCER-2 at TREC 2007. In *Proceedings of the 2007 Text Retrieval Conference (TREC 2007)*, Gaithersburg, Maryland, USA, 2007.

Hirschman, L. and Gaizauskas, R. Natural Language Question Answering: the View from Here. *Natural Language Engineering*, 7(4):275–300, 2001.

Hirst, G. Semantic Interpretation and Ambiguity. *Artifical Intelligence*, 34:131–177, 1988.

Ide, N. and Véronis, J. Word Sense Disambiguation: The State of the Art. *Computational Linguistics*, 24(1):1–41, 1998.

Jones, K. S. Natural Language Processing: A Historical Review. In *Current Issues in Computational Linguistics: In Honour of Don Walker*, pages 3–16. Springer Netherlands, 1994.

Jongejan, B. and Dalianis, H. Automatic Training of Lemmatization Rules that Handle Morphological Changes in Pre-, In- and Suffixes Alike. In *Proceedings of the 47th Annual Meeting of the ACL and the 4th IJCNLP of the AFNLP*, pages 145–153, Suntec, Singapore, 2009.

Jurafsky, D. and Martin, J. H. *Speech and Language Processing*. Pearson Education International, Inc., Upper Saddle River, New Jersey, USA, 2nd edition, 2008.

Khoo, C. and Na, J.-C. Semantic Relations in Information Science. *Annual Review of Information Science and Technology*, 40:157–228, 2006.

Kibble, R. Complement Anaphora and Dynamic Binding. In *Proceedings from Semantics and Linguistic Theory VII (SALT VII)*, pages 258–275, Ithaca, New York, USA, 1997.

Kolomiyets, O. and Moens, M.-F. A Survey on Question Answering Technology from an Information Retrieval Perspective. *Information Sciences*, 181(24):5412–5434, December 2011.

Koo, T., Carreras, X., and Collins, M. Simple Semi-supervised Dependency Parsing. In *Proceedings of the ACL-08: HTL*, pages 595–603, Columbus, Ohio, June 2008. Association for Computational Linguistics.

Kozareva, Z. Bootstrapping Named Entity Recognition with Automatically Generated Gazetteer Lists. In *Proceedings of the 11th Conference of the European Chapter of the Association for Computational Linguistics: Student Research Workshop*, pages 15–21, Trento, Italy, 2006. Association for Computational Linguistics.

Lesk, M. Automatic Sense Disambiguation Using Machine Readable Dictionaries: How to Tell a Pine Cone from an Ice Cream Cone. In *Proceedings the 5th Annual International Conference on Systems Documentation (SIGDOC '86)*, pages 24–26, Toronto, Ontario, Canada, 1986. ACM.

Lovins, J. B. Development of a Stemming Algorithm. *Mechanical Translation and Computational Linguistics*, 11(1, 2):22–31, 1968.

Magnini, B., Vallin, A., Ayache, C., Erbach, G., Peñas, A., de Reijke, M., Rocha, P., Simov, K., and Sutcliffe, R. Overview of the CLEF 2004 Multilingual Question Answering Track. In Peters, C., Clough, P., Gonzalo, J., Jones, G. J., Kluck, M., and Magnini, B., editors, *Multilingual Information Access for Text, Speech and Images*, volume 3491 of *LCNS*, pages 371–391, Berlin Heidelberg, 2005. Springer-Verlag.

Magnini, B., Giampiccolo, D., Forner, P., Ayache, C., Jijkoun, V., Osenova, P., Peñas, A., Rocha, P., Sacaleanu, B., and Sutcliffe, R. Overview of the CLEF 2006 Multilingual Question Answering Track. In Peters, C., Clough, P., Gey, F. C., Karlgren, J., Magnini, B., Oard, D. W., de Rijke, M., and Stempfhuber, M., editors, *Evaluation of Multilingual and Multi-modal Information Retrieval*, volume 4730 of *LCNS*, pages 223–256, Berlin Heidelberg, 2007. Springer-Verlag.

Mausam, Schmitz, M., Bart, R., Soderland, S., and Etzioni, O. Open Language Learning for Information Extraction. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP/CoNLL '12)*, pages 523–534. Association for Computational Linguistics, 2012.

Maybury, M. Toward a Question Answering Roadmap. Technical report, The MITRE Corporation (Information Technology Division), Bedford, Massachusetts, USA, 2003. AAAI Technical Report SS-03-07.

McCandless, M., Hatcher, E., and Gospodnetić, O. *Lucene in Action*. Manning Publications Co., 2010. ISBN 978-1-933988-17-7.

McDonald, R., Pereira, F., Ribarov, K., and Hajič, J. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of the Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing (HLT/EMNLP)*, pages 523–530, 2005.

Mendes, A., Coheur, L., Mamede, N. J., Romão, L., Loureiro, J., Ribeiro, R., Batista, F., and de Matos, D. M. QA@L^{2}F@QA@CLEF. In *Cross Language Evaluation Forum: Working Notes — CLEF 2007 Workshop*, Budapest, Hungary, September 2007. Springer.

Mendes, A., Coheur, L., Mamede, N. J., Ribeiro, R. D., de Matos, D. M., and Batista, F. QA@L^{2}F, First Steps at QA@CLEF. In *Proceedings of the 8th Workshop of the Cross-Language Evaluation Forum (CLEF 2007)*, Lecture Notes in Computer Science, pages 356–363. Springer-Verlag, September 2008.

Missier, P., Romanovsky, A., Miu, T., Pal, A., Daniilakis, M., Garcia, A., Cedrim, D., and da Silva Sousa, L. Tracking Dengue Epidemics using Twitter Content Classification and Topic Modelling. In *Proceedings of the 16th International Conference on Web Engineering (ICWE 2016) / Current Trends in Web Engineering — ICWE 2016 International Workshops*, volume 9881 of *Lecture Notes in Computer Science*, pages 80–92, Lugano, Switzerland, June 2016. Springer-Verlag.

Mitkov, R. *Anaphora Resolution*. Pearson Education, London, UK, 2002.

Mitkov, R., Evans, R., Orasan, C., Barbu, C., Jones, L., and Sotirova, V. Coreference and Anaphora: Developing Annotating Tools, Annotated Resources and Annotation Strategies. In *Proceedings of the Discourse, Anaphora and Reference Resolution Conference (DAARC 2000)*, pages 49–58, Lancaster, UK, 2000.

Moens, M.-F. *Information Extraction: Algorithms and Prospects in a Retrieval Context*. Springer-Verlag, Berlin Heidelberg, 2006.

Mollá, D. and Vicedo, J. L. Question Answering in Restricted Domains: An Overview. *Computational Linguistics*, 33(1):41–61, March 2007.

Mota, C. Resultados Págicos: Participação, Resultados e Recursos. *LinguaMÁTICA*, 4 (1), April 2012.

Mota, C., Simões, A., Freitas, C., Costa, L. F., and Santos, D. Págico: Evaluating Wikipedia-based Information Retrieval in Portuguese. In *Proceedings of the 8th Language Resources and Evaluation Conference (LREC 2012)*, pages 2015–2022, Istanbul, Turkey, May 2012.

Nadeau, D. and Sekine, S. A Survey of Named Entity Recognition and Classification. *Linguisticae Investigationes*, 30(1):3–26, 2007.

Navigli, R. Word Sense Disambiguation: A Survey. *ACM Computing Surveys*, 41(2): 10:1–10:69, 2009.

Navigli, R. and Velardi, P. Structural Semantic Interconnection: a Knowledge-Based Approach to Word Sense Disambiguation. In *SENSEVAL 3: Third International Workshop on the Evaluation of Systems for the Semantic Analisys of Text*, pages 179–182, 2004.

Nicolai, G. and Kondrak, G. Leveraging Inflection Tables for Stemming and Lemmatization. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, pages 1138–1147, Berlim, Germany, August 2016. Association for Computational Linguistics.

Nivre, J., Hall, J., Nilsson, J., Chanev, A., Eryiğit, G., Kübler, S., Marinov, S., and Marsi, E. MaltParser: A Language-Independent System for Data-Driven Dependency Parsing. *Natural Language Engineering*, 13(2):95–135, January 2007.

Oliveira Alves, A., Rodrigues, R., and Gonçalo Oliveira, H. ASAPP: Alinhamento Semântico Automático de Palavras Aplicado ao Português. *LinguaMÁTICA*, 8(2):43–58, December 2016.

Pardo, T. A. S. GistSumm: Um Sumarizador Automático Baseado na Idéia Principal de Textos. Série de Relatórios do Núcleo Interinstitucional de Linguística Computacional, Instituto de Ciências Matemáticas e de Computação, Universidade de São Paulo, São Carlos, São Paulo, Brazil, September 2002.

Pardo, T. A. S., Rino, L. H. M., and das Graças Volpe Nunes, M. GistSumm: A Summarization Tool Based on a New Extractive Method. In *Proceedings of the 6th International Conference on Computational Processing of the Portuguese Language (PROPOR 2003)*, number 2721 in Lecture Notes in Computer Science, pages 210–218, 2003.

Peñas, A., Forner, P., Rodrigo, Á., Sutcliffe, R., Forăscu, C., and Mota, C. Overview of ResPubliQA 2010: Question Answering Evaluation over European Legislation. In *CLEF 2010 LABs and Workshops, Notebook Papers*, Berlin, Heidelberg, New York, 2010a. Springer-Verlag.

Peñas, A., Forner, P., Sutcliffe, R., Rodrigo, Á., Forăscu, C., Alegria, I., Giampiccolo, D., Moreau, N., and Osenova, P. Overview of ResPubliQA 2009: Question Answering Evaluation over European Legislation. In *Proceedings of the 10th Cross-language*

Evaluation Forum Conference on Multilingual Information Access Evaluation: Text Retrieval Experiments, volume 5706 of *LCNS*, pages 174–196, Berlin Heidelberg, 2010b. Springer-Verlag.

Pérez, J., Arenas, M., and Gutierrez, C. Semantics and Complexity of SPARQL. *ACM Transactions on Database Systems*, 34(3):16:1–16:45, August 2009.

Piskorski, J. and Yangarber, R. Information Extraction: Past, Present and Future. In *Multi-source, Multilingual Information Extraction and Summarization*, chapter 2, pages 23–49. Springer-Verlag, Berlin Heidelberg, 2013.

Poibeau, T. and Kosseim, L. Proper Name Extraction from Non-Journalistic Texts. In Daelemans, W., Sima'an, K., Veenstra, J., and Zavrel, J., editors, *Proceedings of the 11th Computational Linguistics in the Netherlands Meeting (CLIN 2001)*, pages 144–157, New York, USA, 2001. Editions Rodopi B.V., Amsterdam.

Prager, J., Brown, E., Coden, A., and Radev, D. Question-Answering by Predictive Annotation. In *Proceedings of the 23rd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 184–191, New York, USA, July 2000a. ACM Press.

Prager, J., Radev, D., Brown, E., Coden, A., and Samn, V. The Use of Predictive Annotation for Question Answering in TREC8. In *Proceedings of the 8th Text REtrieval Conference (TREC 8)*, pages 399–410, Gaithersburg, Maryland, USA, November 2000b.

Prager, J., Radev, D., and Czuba, K. Answering What-Is Questions by Virtual Annotation. In *Proceedings of the 1st International Conference on Human Language Technology Research*, pages 1–5, Morristown, New Jersey, USA, 2001. Association for Computational Linguistics.

Prager, J. M., Chu-Carroll, J., Brown, E. W., and Czuba, K. Question Answering by Predictive Annotation. In *Advances in Open Domain Question Answering*, pages 307–347. Springer-Verlag, Berlin Heidelberg, 2006.

Prolo, C. A., Quaresma, P., Rodrigues, I., Salgueiro, P., and Vieira, R. A Question-Answering System for Portuguese. In *Proceedings of the Workshop on Knowledge and Reasoning for Answering Questions (KRAQ '05 — IJCAI 2005 Workshop)*, Edinburgh, Scotland, UK, July 2005. ACM Press.

Quaresma, P. and Rodrigues, I. A Logic Programming Based Approach to the QA@CLEF05 Track. In *Cross Language Evaluation Forum: Working Notes for the CLEF 2005 Workshop*, 2005a.

Quaresma, P. and Rodrigues, I. A Question-Answering System for Portuguese Juridical Documents. In *Proceedings of the 10th International Conference on Artificial Intelligence and Law*, pages 256–257, New York, USA, June 2005b. ACM Press.

Quaresma, P., Quintano, L., Rodrigues, I., Saias, J., and Salgueiro, P. The University of Évora approach to QA@CLEF-2004. In *CLEF 2004 Working Notes*, 2004.

Ratnaparkhi, A. A Maximum Entropy Model for Part-Of-Speech Tagging. In *Proceedings of Emperical Methods in Natural Langauge Processing*, pages 133–142, Philadelphia, PA, USA, 1996.

Rinaldi, F., Dowdall, J., Hess, M., Mollá, D., Schwitter, R., and Kaljurand, K. Knowledge-Based Question Answering. In *Proceedings of the 7th International Conference on Knowledge-Based Intelligent Information and Engineering Systems (KES 2003)*, pages 785–792, Oxford, UK, September 2003. Springer-Verlag.

Rocha, P. A. and Santos, D. CETEMPúblico: Um Corpus de Grandes Dimensões de Linguagem Jornalística Portuguesa. In *Actas do V Encontro para o Processamento Computacional da Língua Portuguesa Escrita e Falada (PROPOR 2000)*, pages 131–140, Atibaia, São Paulo, Brazil, November 2000.

Rodrigues, R. and Gomes, P. RAPPORT — A Portuguese Question-Answering System. In Pereira, F. C., Machado, P., Costa, E., and Cardoso, A., editors, *Progress in Artificial Intelligence — 17th Portuguese Conference on Artificial Intelligence (EPIA 2015)*, volume 9273 of *Lecture Notes in Computer Science*, pages 771–782, Coimbra, Portugal, September 2015. Springer.

Rodrigues, R. and Gomes, P. Improving Question-Answering for Portuguese using Triples Extracted from Corpora. In *Proceedings of the 12th International Conference on Computational Processing of the Portuguese Language (PROPOR 2016)*, volume 9727 of *LNCS/LNAI*, pages 25–37, Tomar, Portugal, July 2016. Springer.

Rodrigues, R., Gonçalo Oliveira, H., and Gomes, P. Uma Abordagem ao Págico baseada no Processamento e Análise de Sintagmas dos Tópicos. *LinguaMÁTICA*, 4(1):31–39, April 2012.

Rodrigues, R., Gonçalo Oliveira, H., and Gomes, P. LemPORT: a High-Accuracy Cross-Platform Lemmatizer for Portuguese. In Pereira, M. J. V., Leal, J. P., and Simões, A., editors, *Proceedings of the 3rd Symposium on Languages, Applications and Technologies (SLATE '14)*, OpenAccess Series in Informatics, pages 267–274, Germany, June 2014. Schloss Dagstuhl — Leibniz-Zentrum für Informatik, Dagstuhl Publishing.

Russell, S. and Norvig, P., editors. *Artificial Intelligence: A Modern Approach*. Pearson Education, Upper Saddle River, New Jersey, USA, 2nd edition, 2003.

Sager, N. Syntactic Analysis of Natural Language. *Advances in Computers*, 8:153–188, 1967.

Saias, J. and Quaresma, P. The Senso Question Answering Approach to Portuguese QA@CLEF-2007. In Nardi, A. and Peters, C., editors, *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary, September 2007.

Saias, J. and Quaresma, P. The Senso Question Answering System at QA@CLEF 2008. In *Working Notes for the CLEF 2008 Workshop*, Aarhus, Denmark, September 2008.

Sanderson, M. and Croft, W. B. The History of Information Retrieval Research. *Proceedings of the IEEE*, 100:1444–1451, May 2012.

Santos, D. Porquê o Págico? *LinguaMÁTICA*, 4(1), April 2012.

Santos, D. and Bick, E. Providing Internet Access to Portuguese Corpora: the AC/DC Project. In *Proceedings of the 2nd International Conference on Language Resources and Evaluation*, LREC 2000, pages 205–210, 2000.

Santos, D. and Rocha, P. Evaluating CETEMPúblico, a free resource for Portuguese. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics*, pages 442–449, Toulouse, France, July 2001. Association for Computational Linguistics.

Santos, D. and Rocha, P. The Key to the First CLEF in Portuguese: Topics, Questions and Answers in CHAVE. In *Proceedings of the 5th Workshop of the Cross-Language Evaluation Forum*, pages 821–832, Bath, United Kingdom, September 2005. Springer-Verlag.

Sarmento, L. Hunting Answers with RAPOSA (FOX). In *Cross Language Evaluation Forum: Working Notes for the CLEF 2006 Workshop (CLEF 2006)*, Alicante, Espanha, September 2006.

Sarmento, L. and Oliveira, E. Making RAPOSA (FOX) Smarter. In Nardi, A. and Peters, C., editors, *Working Notes for the CLEF 2007 Workshop*, Budapest, Hungary, September 2007.

Sarmento, L., Teixeira, J., and Oliveira, E. Assessing the Impact of Thesaurus-Based Expansion Techniques in QA-Centric IR. In *Proceedings of the 9th Cross-Language Evaluation Forum Conference on Evaluating Systems for Multilingual and Multimodal Information Access (CLEF 2008)*, pages 325–332, Aarhus, Denmark, September 2008a.

Sarmento, L., Teixeira, J., and Oliveira, E. Experiments with Query Expansion in the RAPOSA (FOX) Question Answering System. In *Proceedings of the 9th Workshop of the Cross-Language Evaluation Forum (CLEF 2008)*, Aarhus, Denmark, 2008b.

Savoy, J. Light Stemming Approaches for the French, Portuguese, German and Hungarian Languages. In *Proceedings of the 2006 ACM Symposium on Applied Computing (SAC '06)*, pages 1031–1035, New York, NY, USA, 2006. ACM.

Schlaefer, N., Gieselmann, P., Schaaf, T., and Waibel, A. A Pattern Learning Approach to Question Answering within the Ephyra Framework. In *Proceedings of the 9th International Conference on Text, Speech and Dialogue*, pages 687–694, Brno, Czech Republic, 2006.

Schmid, H. Probabilistic Part-of-Speech Tagging Using Decision Trees. In *International Conference on New Methods in Language Processing*, pages 44–49, Manchester, UK, 1994.

Shrihari, R. and Li, W. A Question Answering System Supported by Information Extraction. In *Proceedings of the 6th Conference on Applied Natural Language Processing*, pages 166–172, 2000.

Simmons, R. F. Answering English Questions by Computer: A Survey. *Communications of the ACM*, 8(1):53–70, January 1965.

Simmons, R. F. Natural Language Question-Answering Systems: 1969. *Computational Linguistics*, 13(1):15–30, January 1970.

Simões, A., Costa, L., and Mota, C. Tirando o Chapéu à Wikipédia: A Coleção do Págico e o Cartola. *LinguaMÁTICA*, 4(1):19–30, April 2012.

Simões, A. M. and Almeida, J. J. jSpell.pm — Um Módulo de Análise Morfológica para Uso em Processamento de Linguagem Natural. In *Actas da Associação Portuguesa de Linguística*, pages 485–495, 2001.

Singhal, A. Modern Information Retrieval: A Brief Overview. *Bulletin of the Technical Committee on Data Engineering*, 24(4):35–43, December 2001.

Snow, R., Jurafsky, D., and Ng, A. Y. Learning Syntactic Patterns for Automatic Hypernym Discovery. In *Advances in Neural Information Processing Systems*, volume 17, pages 1297–1304, 2005.

Soergel, D. The Rise of Ontologies or the Reinvention of Classification. *Journal of the American Society for Information Science*, 50(12):1119–1120, October 1999.

Strzalkowski, T. and Harabagiu, S., editors. *Advances in Open Domain Question Answering*, volume 32 of *Text, Speech and Language Technology*. Springer-Verlag, Berlin Heidelberg, 2006.

Szarvas, G., Farkas, R., and Ormándi, R. Improving a State-of-the-Art Named Entity Recognition System Using the World Wide Web. In Perner, P., editor, *Proceedings of the 7th Industrial Conference on Data Mining (ICDM 2007)*, pages 163–172, Berlin Heidelberg, 2007. Springer-Verlag.

Turing, A. M. Computing Machinery and Intelligence. *Mind: A Quarterly Review of Psychology and Philosophy*, LIX(236):433–460, October 1950.

Unger, C., Bühmann, L., Lehmann, J., Ngomo, A.-C. N., Gerber, D., and Cimiano, P. Template-based Question Answering over RDF Data. In *Proceedings of the 21st International Conference on World Wide Web (WWW '12)*, pages 639–648, Lyon, France, April 2012. ACM Press.

Vallin, A., Magnini, B., Giampiccolo, D., Aunimo, L., Ayache, C., Osenova, P., Peñas, A., de Rijke, M., Sacaleanu, B., Santos, D., and Sutcliffe, R. Overview of the CLEF 2005 Multilingual Question Answering Track. In Peters, C., Gey, F. C., Gonzalo, J., Müller, H., Jones, G. J., Magnini, M. K. B., and de Rijke, M., editors, *Accessing Multilingual Information Repositories*, volume 4022 of *LCNS*, pages 307–331, Berlin Heidelberg, 2006. Springer-Verlag.

Wijaya, D. T. and Mitchell, T. Mapping Verbs In Different Languages to Knowledge Base Relations using Web Text as Interlingua. In *Proceedings of the 15th Annual Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL)*, 2016.

Willet, P. The Porter Stemming Algorithm: Then and Now. *Program: Electronic Library and Information Systems*, 40(3):219–223, 2006.

Winiwarter, W. Syntactic Analysis for Natural Language Interfaces — The Integrated Deductive Approach. In Rowles, C., Liu, H., and Foo, N. Y., editors, *Proceedings of the 6th Australian Joint Conference on Artificial Intelligence*, River Edge, New Jersey, USA, 1993. World Scientific.

Winiwarter, W. and Tjoa, A. M. Morphological Analysis in Integrated Natural Language Interfaces to Deductive Databases. In Matsumoto, Y., editor, *Proceedings of the 4th International Workshop on Natural Language Understanding and Logic Programming*, Nara, Japan, 1993. Nara Institute of Science and Technology.

Wu, F. and Weld, D. S. Open Information Extraction using Wikipedia. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 118–127, Uppsala, Sweden, July 2010. Association for Computational Linguistics.

Xavier, C. C., de Lima, V. L. S., and Souza, M. Open Information Extraction Based on Lexical-Syntactic Patterns. In *Proceedings of the 2nd Brazilian Conference on Intelligent Systems (BRACIS-13)*, pages 189–194, Fortaleza, Brazil, October 2013.

Zhila, A. and Gelbukh, A. Comparison of Open Information Extraction for English and Spanish. *Computational Linguistics and Intelligent Technologies*, 12(19):714–722, 2013.