UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

When the Answer comes into Question in Question-Answering

Ana Cristina Bastos Mendes

Supervisor: Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-Supervisor: Doctor Gábor Magyar

Thesis approved in public session to obtain the PhD Degree in
Information Systems and Computer Engineering

Jury final classification: Pass with Merit

Jury

Chairperson: Chairman of the IST Scientific Board

Members of the Committee:
Doctor Mário Jorge Costa Gaspar da Silva
Doctor Gábor Magyar
Doctor Anselmo Peñas
Doctor Paulo Miguel Torres Duarte Quaresma
Doctor Pável Pereira Calado
Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur

2013

UNIVERSIDADE DE LISBOA
INSTITUTO SUPERIOR TÉCNICO

When the Answer comes into Question in Question-Answering

Ana Cristina Bastos Mendes

Supervisor: Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-Supervisor: Doctor Gábor Magyar

Thesis approved in public session to obtain the PhD Degree in
Information Systems and Computer Engineering

Jury final classification: Pass with Merit

Jury

Chairperson: Chairman of the IST Scientific Board

Members of the Committee:

Doctor Mário Jorge Costa Gaspar da Silva, Professor Catedrático
do Instituto Superior Técnico, da Universidade de Lisboa
Doctor Gábor Magyar, Associate Professor,
Budapest University of Technology and Economics, Hungary
Doctor Anselmo Peñas, Associate Professor,
Universidad Nacional de Educación a Distancia,
Escuela Técnica Superior de Ingenieria Informática, Espanha
Doctor Paulo Miguel Torres Duarte Quaresma, Professor Associado (com Agregação)
da Universidade de Évora
Doctor Pável Pereira Calado, Professor Auxiliar
do Instituto Superior Técnico, da Universidade de Lisboa
Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur, Professora Auxiliar
do Instituto Superior Técnico, da Universidade de Lisboa

Funding Institutions
Fundação para a Ciência e a Tecnologia

2013

Resumo

Um sistema de resposta automática a perguntas tem como objectivo devolver respostas correctas a perguntas formuladas em língua natural. A arquitectura típica deste tipo de sistemas inclui um componente dedicado à Extracção da Resposta, no qual esta tese se foca.

Este documento começa por apresentar o sistema JUST.ASK, desenvolvido no curso destes estudos de doutoramento e que constitui a baseline e o foco de todas as experiências. Seguidamente, aborda as tarefas de extracção de respostas candidatas e de selecção da resposta final. A primeira utiliza padrões lexico-sintácticos, aprendidos automaticamente a partir de fontes de informação de larga escala, através de uma abordagem com supervisão mínima. Na segunda, relações semânticas – equivalência e inclusão – são detectados entre as respostas candidatas e usadas para melhor escolher a resposta final.

Finalmente, é apresentada uma nova abordagem ao problema de responder automaticamente a perguntas, tornando o JUST.ASK no primeiro sistema, de acordo com o nosso melhor conhecimento, que usa as respostas correctas a perguntas anteriores para responder a perguntas futuras, utilizando os padrões entretanto aprendidos. Nesta abordagem, o utilizador tem um papel fundamental, permitindo que o processo de aprendizagem se desencadeie: confirma a correcção da(s) resposta(s) dada(s) pelo sistema. Caso nenhuma das respostas devolvidas esteja correcta, poderá ele próprio fornecer a resposta correcta à pergunta formulada.

Abstract

A Question-Answering system aims at returning correct answers to questions posed in natural language. The typical architecture of a Question-Answering system includes a component dedicated to Answer Extraction, which is the focus of this thesis.

This thesis starts by presenting JUST.ASK, the Question-Answering system developed during the course of these Ph.D. studies, that represents the baseline and the target of all the experiments. Afterwards, the tasks of candidate answer extraction and final answer selection are addressed. The former uses (lexico-syntactic) patterns, automatically learned from large-scale information sources using a minimally supervised approach. In the latter, semantic relations – equivalence and inclusion – are detected between the candidate answers and used to better choose the final answer.

Finally, a new approach to the Question-Answering challenge is presented, making JUST.ASK the first system, to the best of our knowledge, which uses the correct answers to past questions to answer future questions, employing the learned patterns. In this approach, the user has a fundamental role that allows the learning process to be triggered: s/he confirms the correctness of the system's answer(s). However, if none of the returned answers is correct, the user can provide the correct answer to the posed question.

Palavras-Chave

Palavras-Chave

Sistemas de Pergunta/Resposta

Extracção de Respostas Candidatas

Padrões Lexico-sintácticos

Aprendizagem de Padrões

Unificação Padrão/Frase

Estratégias de Relaxamento

Selecção da Resposta Final

Relações Semânticas

Aprender a Responder

Feedback do Utilizador

Keywords

Keywords

Question-Answering Systems

Candidate Answer Extraction

Lexico-syntactic Patterns

Pattern Learning

Pattern/Sentence Unification

Relaxation Strategies

Final Answer Selection

Semantic Relations

Learning to Answer

User Feedback

Acknowledgements

Acabou-se! Ou então é aqui que esta tese tem início... E agora que termina (ou começa), os agradecimentos são devidos:

À minha orientadora, a Prof. Luísa Coheur, por sabiamente (e numa segunda vez) me ter guiado pelos caminhos intrincados do Processamento da Língua Natural, por me motivar em momentos de desânimo, por me dar ideias e por confiar nas minhas e em mim. E à minha amiga, a Luísa, por ser uma fonte de inspiração, por partilhar histórias e dar conselhos, por se rir comigo (*É a partir de agora que vou ser considerada uma crescida?*).

Ao meu grupo, o L^2F, por me ter proporcionado um ambiente de trabalho fantástico e por me ter sempre feito sentir em casa (tanto, que muitos foram os fins-de-semana que passei no 2^o andar do edifício do INESC-ID Lisboa!). Em especial ao Prof. Serralheiro, com quem partilhei sala (e bem me lembro do meu sentimento de quase petrificação nos primeiros dias!) e que tinha sempre qualquer coisa nova e interessante para me ensinar. Ao meu grupo de QA, ao Sérgio, ao Hugo, ao Pedro, ao outro Pedro, ao André, ao Ricardo, por serem excelentes colegas de trabalho.

À Vanda, pela amizade incondicional que me transmite a cada abraço. E à Susana e à Cátia. Às três, pela amizade, pelo apoio, pelas gargalhadas, pela presença que não dispenso, por se terem ido despedir de mim ao aeroporto. À IdRdS... ([Sentir agora arrepio de puro terror, apesar de não se saber muito bem qual a sua origem!]).

Ao meu *sócio*, o Luís Sarmento, que foi um companheiro inesperado de casa e de tango, e que se tornou assim numa espécie de irmão mais velho e experiente. E à Anabela, pelas nossas conversas à porta das nossas casas e que me sabiam sempre tão bem.

Aos que aqueceram o meu Inverno em Pittsburgh (e se esteve frio em Pittsburgh...).

Em particular ao Rui, que me deu as melhores boas-vindas e uma ajuda preciosa na minha integração no burgo (e que esteve sempre atento às novidades do meu blog!), à Laura, que me fez sentir em casa logo assim que passei a porta, à Seza, pela boa disposição contagiante, e ao Avner, um melhor amigo que eu nunca pensei vir a ter (*and just because I thanked you in my thesis, it doesn't mean that I don't hate you!*).

A todas as pessoas importantes que tive a sorte de se terem cruzado no meu caminho e que me ajudaram a tornar-me naquilo que sou hoje. Em especial à Raquel, ao András e ao Dário. E a todos os meus amigos. Sim, mesmo aqueles que, depois de duas teses, continuam a não saber qual é o tópico da minha investigação!

E à minha família. Ao meu irmão. E aos meus pais! Que sempre apoiaram as minhas decisões, mesmo aquelas com as quais não concordavam assim muito. E pelo amor e carinho com que o fizeram. E pronto, Mãe e Pai, prometo que é agora que vou arranjar um emprego "a sério"! Acho eu...

Finalmente, à Fundação para a Ciência e a Tecnologia, que financiou esta tese através da Bolsa de Investigação com a referência SFRH/BD/43487/2008.

Lisboa, Outubro 2013

Ana Cristina Bastos Mendes

Aos meus pais e ao meu irmão

Contents

List of Figures

List of Tables

1 Introduction

The amount of available data grows at a staggering pace. This trend is perhaps most visible in the World Wide Web (WWW), where a large amount of user-generated content (text, audio, video,...) is published every day in social networks, news, blogs and others. In this conjecture, the challenge is to satisfy the user's information needs, regardless of the size of the available data and of its growing rate. The need to deal and leverage those amounts of data to respond to the users' requests led to the foundation of (nowadays) very active research fields, like Data Mining or Information Retrieval, and to the appearance of buzzwords like *big data* [Manning et al., 2008; Baeza-Yates and Ribeiro-Neto, 2011; Rajaraman and Ullman, 2012].

In the particular case of textual data, we have witnessed the rise and establishment of web search engines (like Google[1], Bing[2] or Yahoo![3]) as major role players in people's everyday's lives, supported by the indisputable fact that it is simply impossible for a user to grep such amounts of textual data searching for the information that meets his/her needs. However, it should be clear that oftentimes it is also not viable to let him/her choose among a (possibly) large set of information snippets retrieved by the search engines. Question-Answering (QA) systems appear in this context as the optimal solution to this problem, as they aim to deliver the exact correct responses – answers – to the users' requests – questions posed in natural language. QA systems have been around since the 60's, when they first appeared as natural language interfaces to databases. Since that date, and motivated by the emergence of the WWW, Question-Answering systems began to deal with information sources composed of large amounts of unstructured textual data. The new architecture has becme rather stan-

[1] http://www.google.com/
[2] http://www.bing.com/
[3] http://www.yahoo.com/

dard [Jurafsky and Martin, 2008], consisting of a pipeline composed of three main components, each dedicated to a specific task: the first to the processing of the input question, the second to the retrieval of the textual information where the answers might lie, and the third to the extraction, selection and presentation of the correct answer to the user.

A huge importance is attributed to the understanding of the posed question (the first task in QA), with research lines like question classification being widely explored [Li and Roth, 2002; Pan et al., 2008; Blunsom et al., 2006; Huang et al., 2008; Silva et al., 2011]. The identification of the pieces of information relevant for a question (the second task in QA) has also deserved much attention, and is often based on the developments achieved in the field of Information Retrieval [Tellex et al., 2003; Clarke and Terra, 2003; Khalid and Verberne, 2008; Tiedemann and Mur, 2008]. The primary focus of this Ph.D. thesis lies, however, on the third task within a QA pipeline: the answering task, which we consider as the aggregated tasks of **extracting** candidate answers from the information sources and **selecting** the final answer(s) from the group of candidate answers. Moreover, it will focus on the development of a strategy that allows a QA system to **learn to answer** based on previous, correctly answered questions.

1.1 Thesis Statement

The scientific hypotheses behind this work are the following:

- Relations between the question and answer can be automatically learned and used as an effective approach to **candidate answer extraction**.
- Relations between candidate answers can positively influence the **selection of the final answer** from a set of candidate answers to a question.
- If a QA system is able to assess the correctness of its answers, it can **learn to answer new, previously unseen questions, from past interactions**.

These hypotheses are based on the following.

In what regards the task of candidate answer extraction, some systems [Soubbotin, 2001; Mollá and Gardiner, 2004; Hickl et al., 2006; Shima and Mitamura, 2010] rely on the fact that language comprises several regularities, or patterns, that indicate the existence of certain information. For instance, *The Birth of Venus was painted by Botticelli* is a common way to say that Botticelli made a painting called The Birth of Venus. This is expressed by the sequence of the words *was painted by*. The pattern-based approach to candidate answer extraction relies on the observation that the answer to a given question will probably occur in sentences that contain a rewrite of the original question. For instance, given *Who painted the Birth of Venus?*, the above-mentioned sentence *The Birth of Venus was painted by Botticelli* is composed of a rewrite of the question (*The Birth of Venus was painted*) followed by a preposition (*by*) and the question's correct answer (*Botticelli*). Being so, in this thesis we aim at automatically capturing the patterns that convey the relation existing between (parts of) the question and its correct answer – in the example (*Who painted the Birth of Venus?*, *Botticelli*) this is the relation between the action "painted", the object of the action "the Birth of Venus" and the subject of the action "Botticelli" (which is the answer to the question).

In what concerns the task of final answer selection, a usual approach is the selection of the final answer depending on its frequency of occurrence in the information sources [Brill et al., 2001; Roussinov et al., 2005; Lin, 2007]. However, in many situations, the candidate answers to a question are related among them, which can help to more accurately select the final correct answer. For example, given the question *When was the Birth of Venus painted?* and a group of candidate answers {*1485-86*, *XV century*, *1485*, *1500*, *c1485*}, we aim at relating them (`equals(`*1485*`,`*c1485*`)`, `includes(`*XV century*`,`*1485*`)`, `includes(`*1485-1486*`,`*1485*`)`, and so forth) through previously defined semantic relations. Therefore, in this thesis we aim at improving the selection of the final answer by detecting and using the relations that the candidates hold between them.

Finally, a typical interaction with a QA system is that in which a question is posed and its answer is returned. This means that, after being presented to the user, the system's answers are usually discarded from further processing. However, there is much information

conveyed by the correct answer to a question that is simply lost in every interaction. For example, knowing that *Botticelli* correctly answers the question *Who painted The Birth of Venus?* and that it can be found in the sentence *Botticelli has painted The Birth of Venus*, such information might be important to locate the correct answer to the *similar* question *Who wrote the Divine Comedy?*. This problem is addressed in this thesis as we aim at allowing a QA system to learn new patterns (and to answer new questions) using information obtained from positive feedback given in previous interactions.

This thesis will focus on open-domain, factoid QA. Factoid QA systems are expected to deliver short, fact-based answers to natural language questions. Since the questions are not limited to any particular domain, rather they can be about virtually anything, systems are told to be open-domain.

1.2 Thesis Contributions

The contributions of this thesis are the following:

- The reimplementation/improvement, description and thorough evaluation of a running open-domain QA system (called JUST.ASK);
- The proposal, description and evaluation of an algorithm based on a minimally supervised approach to learn answer extraction patterns from pairs of questions and their answers;
- The proposal, description and evaluation of a pattern/sentence unification method, which relies on the existence of relaxation strategies that flexibilize that unification;
- The description and evaluation of an approach to final answer selection based on the detection of semantic relations between candidate answers;
- The proposal, description and evaluation of an approach that allows the system to iteratively learn to answer new questions based on previous correct interactions;

- The description of the modifications that occurred in a QA system's architecture in order to support learning from previous interactions.

Moreover, the following tools and resources were also made freely available for the research community:

- JUST.ASK, the open-domain QA system described in the thesis and in a recent journal paper [Mendes et al., 2013b].
- A corpus that consists of 200 questions, their categories, a set of snippets retrieved from the Web and all the correct answers occurring in the snippets.
- The JUST.ASK's question classifier [Silva et al., 2011].

It should be mentioned that JUST.ASK benefitted from collaborations with colleagues from my group, the Spoken Language Systems Laboratory (L^2F)[4] of INESC-ID, Lisboa. In fact, the origins of this system date back to the scholar year of 2008/2009, when a QA system named QA+ML@Wikipedia&Google was built in the context of a Master's Thesis [Silva, 2009]. The goal was the creation of a QA system that would combine established techniques on the Natural Language Processing and Machine Learning fields, and the main efforts were dedicated to question classification.

By the end of 2009, QA+ML@Wikipedia&Google was handled over to me and became my main research tool. My first tasks in the improvement of the system included code refactoring, cleaning and documentation, the indexing of a local document collection, the development of new regular expressions to answer extraction, the development of the answer normalizers, the establishment of a pipeline for answer selection. Also, I focused on the question classification component of the system, which led to the publication of a journal paper [Silva et al., 2011].

After the first improvements, the system was renamed to JUST.ASK and the baseline system for my PhD thesis was settled. JUST.ASK was fully described and thoroughly evaluated

[4] https://www.l2f.inesc-id.pt

in a journal paper [Mendes et al., 2013b], a work which was possible due to creation of an evaluation corpus in collaboration with Gonçalves [2012]. Recent work I have conducted in JUST.ASK has specially addressed extensions and improvements to the Answer Extraction component of the system, which will be fully described in this thesis.

1.3 Research Methodology

The work presented in this thesis was integrated in the QA line of research of L^2F.

The general methodology of the thesis consisted of the following: first, the baseline of the system was established, which allowed to have the first results against which the improvements brought to the system by any strategy could be measured. The baseline (including tools and testing data) was released in order to serve as a benchmark, since there is a shortage of available resources that allow a true comparison of the advancements in this area. Afterwards, the first two scientific hypotheses of this research were tackled individually, since they relate with two self-contained sub-tasks in the QA pipeline: candidate answer extraction and final answer selection. The best configurations regarding both tasks were integrated in the system. Finally, the system's architecture was changed to test the last hypothesis, allowing an iterative approach to QA where new questions are answered based on previous (successful) interactions.

In each of these steps, a thorough evaluation was performed, based on traditional evaluation metrics in the QA field of study (*e.g.*, recall, precision and accuracy). Moreover, regarding the evaluation corpora and information sources, whenever possible we employed data that is freely available and used in the context of evaluation forums of QA systems. During this work, we tried not to commit to any paradigm, and made use of several techniques, from more traditional rule-based to more recent machine learning-based.

All the publications, tools and resources resulting from this thesis are freely available and can be found online at `https://qa.l2f.inesc-id.pt/`.

1.4 Thesis Structure

This thesis does not follow the typical organization of an engineering work presentation, with an introduction/related work/architecture/evaluation/conclusion structure. It is our opinion that the current structure will allow a better understanding by the reader, since in each chapter we focus on a different, self-contained aspect of the developed work, which we introduce with references to related work, describe, evaluate and discuss.

This thesis is structured as follows:

- Chapter 2 describes JUST.ASK, the open-domain QA system that is the focus of this thesis. We detail the pipelined architecture of JUST.ASK, including its Question Processing (in Section 2.2.1), Passage Retrieval (in Section 2.2.2) and Answer Extraction components. We devote particular attention to the latter component (in Section 2.2.3). Moreover, we describe the corpora built to evaluate the system in Section 2.3 and in Section 2.4 the evaluation metrics are presented. The results of the evaluation of the baseline of the system are shown in Section 2.5. We make a brief discussion in Section 2.6 and conclude the chapter in Section 2.7.

- Chapter 3 focuses on the task of candidate answer extraction, using learned (lexico-syntactic) patterns. In Section 3.1, we start by reviewing related work in answer extraction. After, in Section 3.2, we detail a minimally supervised approach to learn patterns that connect questions to their answers and, in Section 3.3, our method to use the patterns to extract correct answers to factoid questions. Next, we present the results of the evaluation of our pattern learning and matching strategies (in Section 3.4). We make a brief discussion in Section 3.5 and conclude the chapter in Section 3.6.

- Chapter 4 addresses the task of final answer selection, in particular in using semantic relations in order to make a better selection of the system's final answer. We start by referring to related work in Section 4.1, namely we present four semantic relations – equivalence, inclusion, aggregation and alternative – and a review of techniques for

their detection. Next, we introduce our approach to answer selection based on the semantic relations (in Section 4.2), including how we detect the relations of equivalence and inclusion between answers (in Section 4.2.1), and we select the final answer based in the relations (in Section 4.2.2). We present the results of the evaluation of the approach in Section 4.3. We make a brief discussion in Section 4.4 and sum up the chapter in Section 4.5.

- Chapter 5 describes our approach that allows JUST.ASK to use the correct answers to previous questions to learn patterns to answer future questions. In Section 5.1 we start by showing approaches that rely on learning from questions and (their) answers. Afterwards, we describe our iterative approach to learn to answer new questions from past successful interactions (Section 5.2.1), we refer to the changes that were made on JUST.ASK to accommodate the work done regarding the candidate answer extraction and final answer selection tasks and the system's new architecture (in Section 5.2.2). In Section 5.3 we evaluate our approach and present JUST.ASK's new results. We discuss the chapter in Section 5.4 and sum up the chapter in Section 5.5.

- The thesis finishes in Chapter 6, where we summarize the document. In Section 6.1, we refer to the contributions of this Ph.D., including the resources made publicly available to the research community and the published contributions (accepted scientific conference and journal papers). Finally, we point to future work directions and we indicate new paths in which this work can evolve (in Section 6.3).

2 Just.Ask: A Multi-pronged Approach to Question-Answering

The present chapter is dedicated to introduce JUST.ASK, its architecture and main components. It starts with a review of related work in QA, followed by the description of the system. A thorough baseline evaluation is also presented, using two different corpora. Afterwards, the evaluation results are analysed and discussed. The chapter ends with a brief summary.

2.1 Related Work

Question-Answering (QA) has deserved a great deal of attention from the research community and, nowadays, it is still a very active field of research. Almost 50 years ago, Simmons [1965] published a survey article describing no less than fifteen QA systems for the English language that were built in the preceding five years. The system `BASEBALL` [Jr. et al., 1961], which handled questions about baseball games played in the American League over a period of one year, was among them. A few years later, with the sponsorship of NASA, William Woods [1972] developed `LUNAR`, which answered questions about lunar rock and soil samples that were collected by the Apollo Moon missions. The system was demonstrated at the Lunar Science Conference in 1971, where it answered correctly to 78% of the questions posed by the attending geologists [Hirschman and Gaizauskas, 2001].

Both `LUNAR` and `BASEBALL` were essentially natural language interfaces to databases (NLIDB), in which the user's question is translated into a database query, and the query's output is returned as the answer. Many other systems were developed along the lines of `LUNAR` and `BASEBALL`; however, the majority of them are limited to restricted-domains, their knowledge is stored in a database and are very complex to port to different domains. A detailed survey on NLIDBs can be found in Androutsopoulos et al. [1995].

With the advent of the WWW in the early 1990s, and the resultant explosion of electronic media, many research groups began to exploit the web as a large text corpus, creating the so called web-based question answering systems, such as START[1] [Katz, 1988, 1997]. In 1999, with the launch of the QA track [Voorhees, 1999] in the renowned TREC[2], the field of QA gained new momentum. In fact, in the recent years, several evaluation forums, like the Cross-Language Evaluation Forum (CLEF) and TREC, have promoted and testified many of the improvements of QA systems. The participating systems implement an extensive panoply of techniques, but the general architecture of a modern information retrieval-based QA system has become somewhat standardized [Jurafsky and Martin, 2008]: it consists of a pipeline composed of three main components dedicated to the tasks of Question Processing, Passage Retrieval and Answer Extraction, respectively. Figure 2.1 depicts the typical QA pipeline.

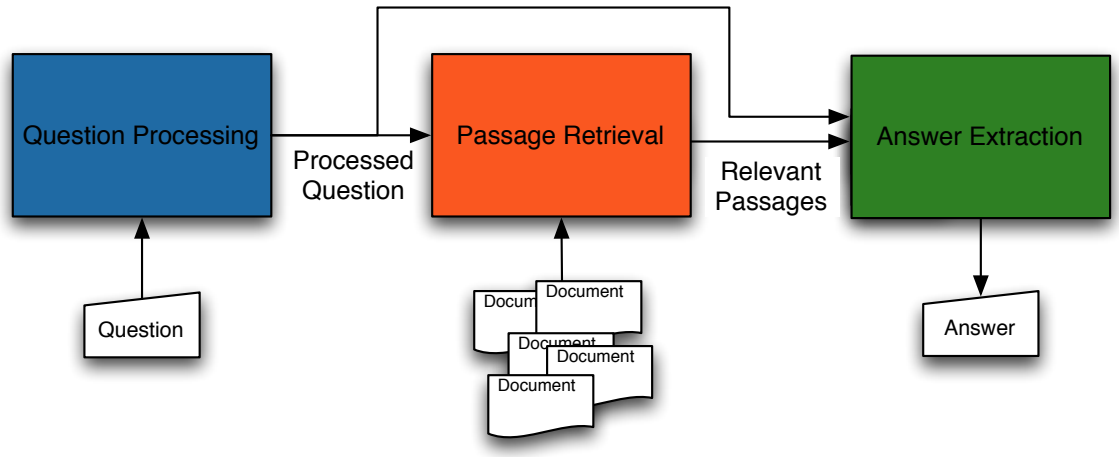


Figure 2.1: Typical pipelined architecture of a QA system.

Question processing (also known as question analysis or question interpretation) is the task responsible for understanding the posed question and often involves several subtasks, like **question classification** or the extraction of different types of information from the question (named entities or main verbs, for example). In question classification (a subtask transversal to most systems), a semantic category representing the type of the answer that is being sought after is assigned to the question. It is considered of extreme importance to a QA system due

[1] http://start.csail.mit.edu/
[2] http://trec.nist.gov/

to two reasons:

1. the classification can help narrowing down the number of possible candidate answers and, depending on the question category, different strategies can be used to extract the correct answer from the information sources; and

2. a misclassified question can hinder the ability of the system to reach the right answer, because it can cause downstream components to make wrong assumptions about the question.

Therefore, to achieve good overall results, it is of crucial importance to have a precise question classifier.[3] The set of question categories is referred to as question type taxonomy and several question type taxonomies have been proposed in the literature [Moldovan et al., 2000; Hermjakob et al., 2002; Li and Roth, 2002]. The hierarchical taxonomy of Li and Roth [2002] is one of the most well known and widely used in QA, mainly due to the fact that, besides the taxonomy, the authors published nearly 6,000 labelled questions and made them freely available. As so, the use of this corpus in the evaluation of machine learning techniques for the purpose of question classification is recurrent in the literature [Li and Roth, 2002; Pan et al., 2008; Blunsom et al., 2006; Huang et al., 2008].[4]

After the question processing, the next step is to find the relevant passages where the answer(s) to the question might be found. This step is usually referred to as **passage retrieval**. It should be noticed that some systems, like the one of Grau et al. [2006], prefer to retrieve documents instead of passages and analyze them when searching for the answer. If, on the one hand, this approach can introduce noisy information to be later filtered in the answer extraction module, on the other hand, the usage of advanced Natural Language Processing (NLP) techniques can potentiate the detection of answers hidden in anaphoric and elliptic expressions, which is more complex (or even impossible) to achieve when using passages. The trade-off between the use of passages and documents as retrieval units has been

[3] For a study on the impact of question classification in QA, refer to the work of Moldovan et al. [2003b]
[4] A more detailed review of question classification can be found in the paper of Silva et al. [2011].

the subject of study of several authors (see Clarke and Terra [2003] and Monz [2003]). Despite such studies and the importance of this step in the overall performance of a QA system, the passage retrieval component is often neglected by QA developers and researchers in the sense that most prefer to rely on pre-existing off-the-shelf solutions, like Google or Lucene[5] search engines. However, there are still a few systems, such as the one from `PRIBERAM` [Amaral et al., 2008], `QA@L2F` [Mendes et al., 2008] or `QRISTAL` [Laurent and Sophie Nègre, 2005], that endeavour in a stage of pre-processing the corpus, by collecting, organizing and storing the information contained in thousands of documents in databases. The task of passage retrieval can take advantage of the output of the preceding question classification task, to retrieve only those documents that contain at least one occurrence of a named entity that agrees with the expected question category. By doing so, the answer extraction module only has to search passages/documents that may potentially have the correct answer. Another fundamental task in passage retrieval has to do with **query formulation**, in which the question is translated into a suitable representation that can be used by the search engine to retrieve relevant passages. Note that this representation is not necessarily an unstructured set of keywords (it can, for instance, be a structured query language) and is tightly coupled and dependent of the used search engine.

The **answer extraction** is the last task in the typical QA pipeline, responsible for extracting and selecting the final answer(s) to the posed question from the relevant passages returned by the passage retrieval component and present it to the user. As an example, consider the question *Who painted the Birth of Venus?*, classified as HUMAN:INDIVIDUAL, and for which the following passages were retrieved:

- *The Birth of Venus is a 1486 painting by **Sandro Botticelli**.*
- *The Birth of Venus: A Novel is a 2003 novel by **Sarah Dunant**, a bestselling British author. The story is set in the late 15th century in Florence, Italy.*
- *The Birth of Venus is probably **Botticelli**'s most famous painting*

[5] `http://lucene.apache.org/`

- ***Alessandro di Mariano di Vanni Filipepi**, better known as **Sandro Botticelli** ... Among his best known works are The Birth of Venus and Primavera.*

Given the above information, the answer extraction task should extract the candidate answers from the passages (the individual names in bold face), select the correct answer (*Sandro Botticelli*) and, finally, present it to the user. In this example are patent some of the challenges that QA systems must circumvent, in particular:

- to realize that a *painting* is the result of the action *painted* (or *to paint*) expressed in the question;
- to realize that the expression *Birth of Venus* can refer either to a novel or a painting. Given that the question mentions *painted*, the candidate answer *Sarah Dunant* is wrong; and,
- to realize that the candidate answers *Sandro Botticelli*, *Botticelli* and *Alessandro di Mariano di Vanni Filipepi* are equivalent.

2.2 Introducing Just.Ask

JUST.ASK is an open source QA system for the English language, entirely implemented in Java and freely available for the research community. The parametrization of JUST.ASK is primarily done in an XML file. Among others, in this file one can parameterize the different components of JUST.ASK, as well as the system general configuration.

2.2.1 Question processing in Just.Ask

The **Question Processing** component of JUST.ASK receives a natural language question as input and outputs a processed question, which is a structure composed of several different types of information extracted from the question, including: the question tokens, the question syntactic components, the question headword, the question focus and the question category.

Therefore, this component is responsible for two main tasks: question analyzis and question classification. For this, it makes use of available NLP tools/resources and techniques. Moreover, JUST.ASK implements a module that, depending on the question, determines its headword or focus. A detailed view of the Question Processing component of JUST.ASK is shown in Figure 2.2.

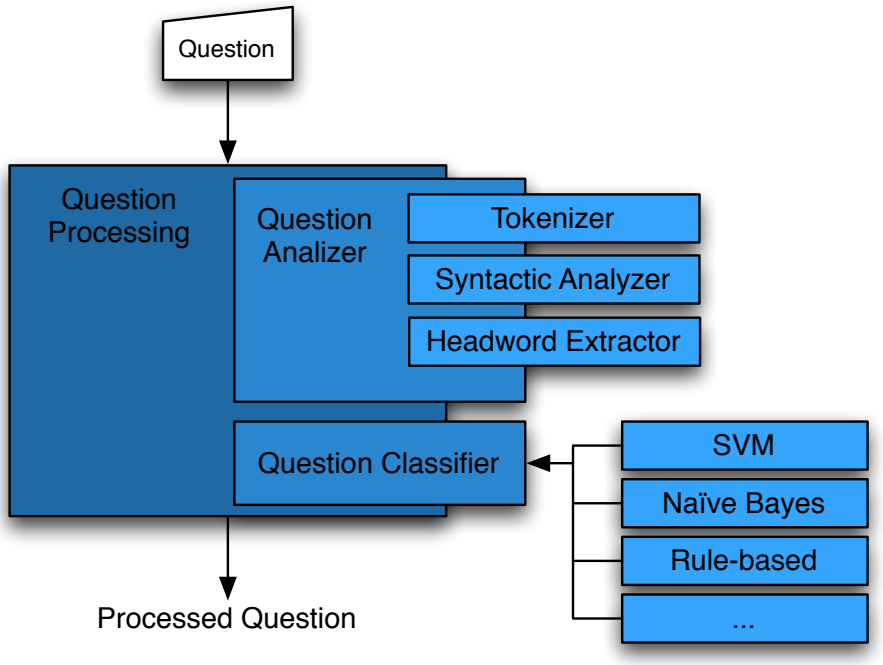


Figure 2.2: Detailed view of the Question Processing component of JUST.ASK.

JUST.ASK classifier exploits the widely used Li and Roth [2002]'s two-layer question type taxonomy, consisting of 6 coarse.grained categories (ABBREVIATION, DESCRIPTION, ENTITY, HUMAN, LOCATION, and NUMERIC), which are further refined into 50 fine-grained categories (see Table 2.1). By using this taxonomy, JUST.ASK's question classifier can be directly compared with other classifiers described in the literature.

The JUST.ASK's classifier permits the usage of different classification techniques. In particular, classification can resort to hand-built rules or machine-learning techniques, namely Support Vector Machines (SVM) and Naïve Bayes. State of the art results were attained by the classifier of JUST.ASK when modeling the task of question classification as a supervised learning classification problem (using SVM), although the most successful features used to

Coarse	Fine
ABBREVIATION	abbreviation, expansion
DESCRIPTION	definition, description, manner, reason
ENTITY	animal, body, color, creative, currency, medical disease, event, food, instrument, language, letter, other, plant, product, religion, sport, substance, symbol, technique, term, vehicle, word
HUMAN	description, group, individual, title
LOCATION	city, country, mountain, other, state
NUMERIC	code, count, date, distance, money, order, other, percent, period, speed, temperature, size, weight

Table 2.1: Li and Roth's two-layer taxonomy for question classification.

train the model are generated by the rule-based classifier. In fact, after several experiments, we realized that the use of unigrams, headwords and its semantic classification as features resulted in the highest accuracy, being the last two set of features created by the rule-based classifier [Silva et al., 2011].

In what concerns the question headword, JUST.ASK adopts a similar notion to that described in Huang et al. [2008]: the headword is a word in a question that *represents* the information that is being sought after. For instance, considering the question *What is the capital of Spain?*, *capital* is its headword. Note that many questions do not have a headword. For example, in *How many plays has Shakespeare written?* one is searching for the number of plays written by the poet, not the plays themselves (for that, the question could be *Which plays has Shakespeare written?*, with headword *plays*). In order to find the headword of a question, JUST.ASK's uses its parse tree, traversed top-down by a pre-defined set of rules – the head-rules. For the purpose of parsing the input question, we use the Berkeley Parser [Petrov and Klein, 2007], trained on the QuestionBank [Judge et al., 2006], a treebank of 4000 parse-annotated questions. When it comes to the head-rules, they are a heavily modified version of those given in the work of Collins [1999], specifically tailored to extract headwords from questions. There are, however, a few exceptions to the head-rules and, for that, we use a set of *non-trivial rules* that determine the correct headword for situations that cannot be covered by the *head-rules*.

In order to enrich the question headword with semantics, WordNet [Fellbaum, 1998], the lexical database of English, was incorporated in JUST.ASK. Therefore, sets of related WordNet synsets were manually grouped into 50 clusters [Silva, 2009], each representing a question category. Then, the question headword was translated into a WordNet synset using a set of heuristics. Finally, a breadth-first search on the translated synset's hypernym tree was employed, in order to find a synset that pertains to any of the pre-defined clusters; several heuristics were used to aid word sense disambiguation.

A thorough description of the classification task in JUST.ASK can be found in the work of Silva et al. [2011], which also includes an extensive report on the experiments we conducted to find the most promising features to train the machine-learning classifier. The set of hand-built rules used in JUST.ASK rule-based classifier is described in the same paper and is publicly available for research purposes at `http://qa.l2f.inesc-id.pt/wiki/index.php/Resources`. At the present moment, JUST.ASK depends on this classifier which, in turn, relies on Li and Roth's question type taxonomy.

2.2.2 Passage retrieval in Just.Ask

The **Passage Retrieval** component of JUST.ASK receives as input the processed question from the preceding component and outputs the set of relevant passages for the posed question.[6] Figure 2.3 shows a zoom-in perspective of this component.

JUST.ASK employs a multi-strategy approach to passage retrieval, with each strategy tailored to a specific question category or groups of question categories. Different strategies involve the use of different **information sources** and different **query formulations**. Once the information source to be used has been selected and the queries have been formulated, the last step is to submit the queries to the information source's endpoint and fetch the results. These queries are submitted and processed in parallel, using multiple threads, in order to reduce the amount of time dedicated to this task. The results retrieved for each query are

[6]Throughout this paper, we will refer to the output of the Passage Retrieval component as *passage*, regardless of it being a paragraph or a short summary (snippet) of a document retrieved by a search engine.

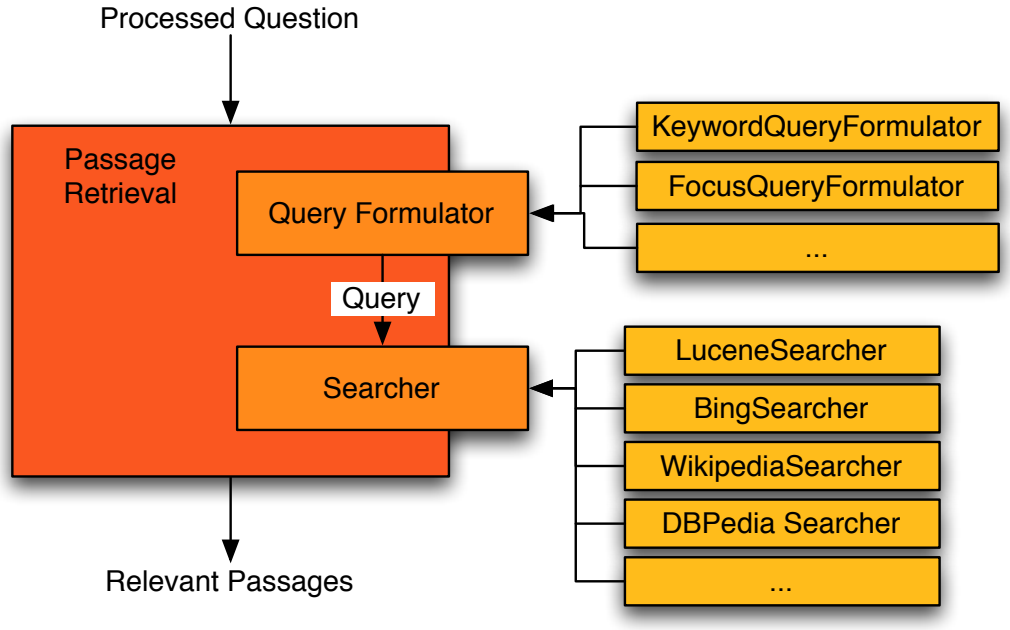


Figure 2.3: Detailed view of the Passage Retrieval component of JUST.ASK.

aggregated (with duplicates removed) and sent to the answer extraction component for further processing. Besides the passages, relevant metadata is collected and stored (being available if necessary), including: the rank of each result is the list of retrieved results, the title and url of the respective webpage/resource, and whether the information source is unstructured or not.

The usage of a certain information source depends on the category attributed to a question: while search engines are typically used to retrieve relevant passages to factoid-like questions, the semi-structured sources (like Wikipedia[7] and DBPedia [Auer et al., 2007]) are used to answer non factoid-like questions that require longer answers.

In JUST.ASK, the inclusion of a new strategy to Passage Retrieval implies modifications in the XML configuration file (besides the addition of the class(es) that implement the strategy). This file also allows to parametrize the search engines in use, respective query formulators and maximum number of passages to be retrieved. However, if one desires to associate a search engine to a specific question category, this must be done programmatically.

[7] http://www.wikipedia.org/

The following subsections describe the information sources and the query formulation strategies currently available in JUST.ASK.

2.2.2.1 Information sources

At the present moment, JUST.ASK uses Lucene search engine and the Bing search API[8] to retrieve relevant passages from unstructured sources; Wikipedia and DBpedia are repositories of semi-structured content also used by the system.

The passages from the search engines (sources of unstructured information) correspond to the snippets/paragraphs that the search engine associates with each returned search result. Due to their encyclopaedic nature, JUST.ASK uses the semi-structured sources to answer non factoid-like questions that require longer answers. In particular, Wikipedia is utilized to answer DESCRIPTION:DEFINITION and HUMAN:DEFINITION questions. DBpedia is used in conjunction with Wikipedia. Briefly, Wikipedia is used to locate the article where the answer to a given question might be found, while DBpedia is used to extract the actual answer from the article in a structured manner, without having to access the full-text of the article's web page. We discuss this strategy in the following subsection.

2.2.2.2 Query formulation

The most simple query formulation strategy is based on keywords and consists of generating a query that comprises all the words in a given question, except the stopwords, question words and punctuation. For instance, given the question *When was Beethoven born?*, the query `Beethoven born` would be generated. The keyword query formulator is used to generate queries for search engines and, therefore, it is applied to every question whose category is associated with this information source – *i.e.*, factoid-like questions.

The simplicity of the query formulation strategy based on keywords is indisputable. However, oftentimes questions contain expressions that should not be modified and should be sent

[8] `https://datamarket.azure.com/dataset/8818F55E-2FE5-4CE3-A617-0B8BA8419F65`

to the search engine *as is* (*e.g.*, questions containing quotes, referring for instance to book or movie titles, or citations). An extended keyword query formulation strategy was built; it only discards words that are not contained in the quote, while those expressions that should be present in the search results are put in between quotation marks in the query.[9]

A last query formulation strategy is based on the focus of the question. The literature in QA is not consensual in the definition of question focus, but in JUST.ASK we adopt a similar notion to that of Bunescu and Huang [2010], who see it as 'the set of all maximum noun phrases in the question that co-refer with the answer'. For example, in the question *Who was Afonso Henriques?*, *Afonso Henriques* is the focus as it co-refers with the answer *first king of Portugal*. This formulator is applied in non factoid-like questions and, therefore, used in a combined strategy that exploits both Wikipedia & DBpedia. The general idea is to use Wikipedia's search to locate the title of the article where the answer to a question might be found, and then use DBpedia to retrieve the *abstract*[10] of the article, which is considered the answer. For that purpose, the focus query formulator builds the query uniquely with the focus of the question. Considering again the example question *Who was Afonso Henriques?*, a query with the question focus *Afonso Henriques* is generated and sent to Wikipedia's API. Second, the first result returned – *Afonso I of Portugal*, in this case – is transformed into a DBpedia resource – `http://dbpedia.org/resource/Afonso_I_of_Portugal`. At last, we create the query to retrieve the abstract of the article from DBpedia.

2.2.3 Answering in Just.Ask

The **Answer Extraction** component receives the processed question originated in the Question Processing component, as well as the relevant passages retrieved by the Passage Retrieval component. The answer extraction can be divided in two stages – **candidate answer extraction** and **final answer selection**. For candidate answer extraction, we take advantage of the rich question type taxonomy utilized in this work to devise strategies for each particular

[9]The syntax of the query often varies according to the search engine in use. Putting one or more words in between quotation marks is the common syntax used to obligate the search results to contain that word/expression.

[10]A abstract in DBPedia roughly corresponds to the first paragraph of the respective Wikipedia article.

question category or groups of question categories. For instance, for NUMERIC type questions, we employ an extensive set of regular expressions to extract candidate answers, whereas for HUMAN type questions, we use a machine learning-based named entity recognizer. In what regards the answer selection, our strategy is to first normalize candidate answers, aggregate answers by lexical equivalence, apply a clustering algorithm to group together similar answers and then filter out unwanted candidate clusters. Finally, since each resulting cluster is scored, the final answer is decided by *ballot* – i.e., the representative answer within the cluster with highest score is chosen.

The Answer Extraction component and its sub components are depicted in Figure 2.4.

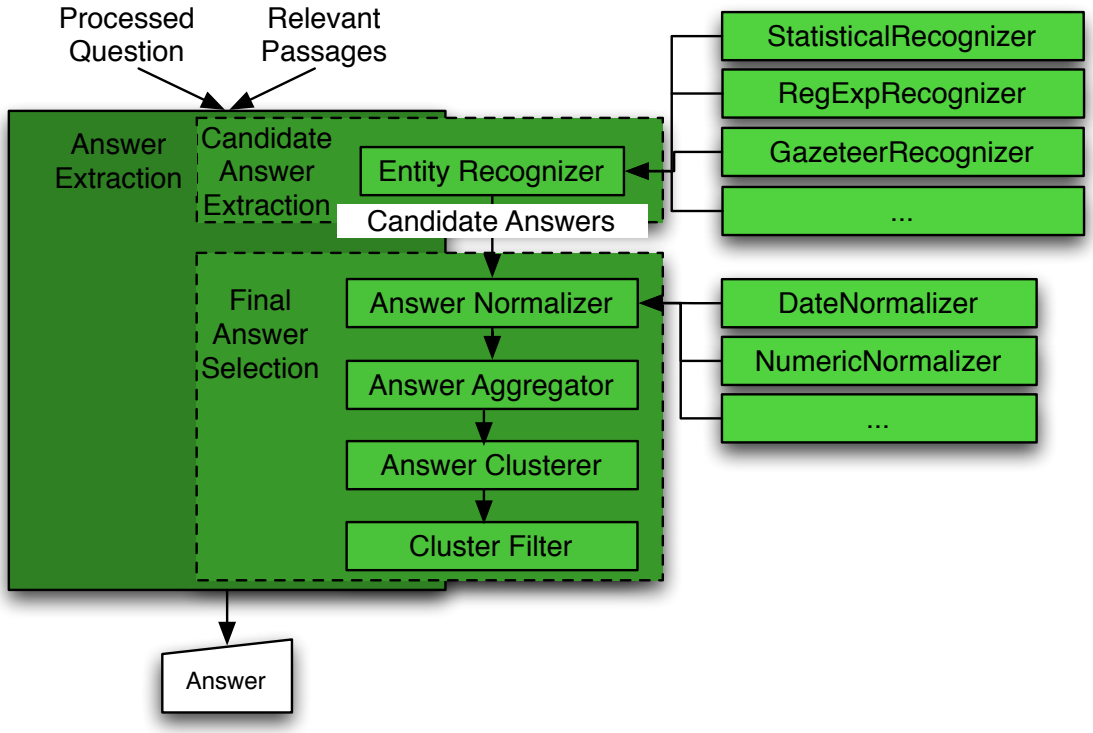


Figure 2.4: Detailed view of the Answer Extraction component of JUST.ASK.

In JUST.ASK, the inclusion of a new strategy to candidate answer extraction implies modifications in three different files (besides the inclusion of the class(es) that implement the strategy): the XML configuration file, which contains reference to the answer extraction strategies active in the current session; the class that keeps track of the list of available answer extraction strategies; and a class that represents a factory of all the available recognizers in

the system. Also in the XML configuration file, one is able to deactivate (or activate) the candidate answer extraction strategies to be used, as well as each of the (optional) tasks that can be performed before the selection of the final answer.

2.2.3.1 Candidate answer extraction

In the following we describe the strategies developed to extract candidate answers from relevant passages (namely, JUST.ASK uses a statistical recognizer for the purpose of Named Entity Recognition (NER), Regular Expressions, Gazetteers, WordNet-based recognizers), as well as the question categories that each strategy is associated with.

An extensive set of regular expressions was created to extract candidate answers for questions of category NUMERIC. Regular expressions provide a very concise way to describe candidate answers for this type. For instance, to identify potential answers to NUMERIC:TEMPERATURE questions, the regular expression `/[0-9]+(K|R|°C|°F)/` could be used. Expressions were also created in a modular manner, with a *numerical basis* being shared among several categories. For example, both NUMERIC:DISTANCE and NUMERIC:TEMPERATURE share the same numerical basis, with the only difference being in the units that follow the numbers – linear measures and temperature units, respectively. Moreover, we developed a set of regular expressions that are made specific to each question, since they are built with the question focus. For instance, given *What does NEO stand for?*, we dynamically create a group of expressions in order to match answers to that question, such as `/.* +(NEO)/` or `/.* +, NEO,/`. Currently, these regular expressions exist uniquely for questions that belong to categories ABBREVIATION and NUMERIC:COUNT, but can be further extended. Each regular expression is paired with a numeric score, which is assigned to every candidate answer that is matched and extracted by it. As a final remark, it is worth mentioning that the regular expressions utilized in this work are far more complex than those of the examples we have given, and take into consideration a wide range of formats and numeric units.

Although regular expressions are a very powerful tool, they can become cumbersome to

use when what we are trying to search for is not in a *rigid* format. For instance, some of the questions require particular names as candidate answers – e.g., HUMAN:INDIVIDUAL questions call for person names –, which can occur in many different formats, and are therefore difficult to express using regular expressions. Moreover, some of these names can refer to different entities, depending on the context in which they occur, thus aggravating the problem. An example of this situation is the name *Washington*, which can either refer to a city, a state, or a person. To cope with the above problems, we used a machine learning-based named entity recognizer, which is able to automatically learn a model to extract entities, based on a set of annotated examples. In particular, we employed Stanford's Conditional Random Field-based named entity recognizer [Finkel et al., 2005], which is able to recognize four entity types: PERSON, LOCATION, ORGANIZATION, and MISCELLANEOUS. The latter serves as a container for named entities that do not fit in the first three categories, such as book and song titles.

JUST.ASK is also able to deal with *type of* questions. Consider the ENTITY:ANIMAL question *Which **animal** is the fastest?*. In this question, the answer is not a particular instance of an animal, but rather a *type of* animal – cheetah. These answers are very difficult to extract from natural language text, since they can be easily confused with other nouns that are present in relevant passages. A new approach was suggested by Silva [2009] to extract answers for *type of* questions, using WordNet's hyponymy relations, which we use in JUST.ASK. Based on the fact that candidate answers for these questions are often hyponyms of the question's headword, a dictionary is constructed in *run time*[11] with the entire hyponym tree of the headword. The dictionary is then used by an exact dictionary matcher algorithm to extract candidate answers. For this work, LingPipe's implementation of the Aho-Corasick [Gusfield, 1997] algorithm was used. Also, as a corollary of this strategy, particular instances of a given word can also be extracted, if they exist in WordNet. For example, in the questions *What is the largest **planet** in the Solar System?* and *What is the world's best selling **cookie**?*, both *Jupiter* (⇒ *planet*) and *Oreo* (⇒ *cookie*) are extracted. This algorithm is utilized in all questions that belong to the ENTITY category, with the exception of ENTITY:EVENT,

[11] We use the term *run time* to refer to the fact that the dictionaries are not constructed *a priori*, but rather when they are needed, in run time.

ENTITY:LETTER, ENTITY:TERM, and ENTITY:WORD. Moreover, it is also used for the categories HUMAN:TITLE, LOCATION:MOUNTAIN, and LOCATION:OTHER.

Finally, certain question categories, such as LOCATION:COUNTRY, have a very limited set of possible answers – names of all the countries in the world, in this case. For these situations, a gazetteer[12] can help [Lita et al., 2004] to accurately extract candidate answers, as it can be used to assure that only candidate answers of the expected type are extracted. We used a gazetteer for both LOCATION:COUNTRY and LOCATION:CITY categories. The gazetteers are utilized in a similar way as the exact dictionary matcher described previously in this section, with the difference being in the fact that gazetteers are not constructed in *run time*, but they already exist when the system starts up.

As a final note, we should mention that the answer extraction strategies are applied in parallel, using multiple threads (one thread per passage), in order to maximize the performance of the system. Moreover, since in essence all of them aim at extracting named entities from text, for reference purposes we denominate this set of strategies as NE-based strategies to candidate answer extraction.

2.2.3.2 Final answer selection

After candidate answers have been extracted, the last step is to choose the final answer to be returned. In JUST.ASK, four tasks can be performed before this decision: **normalization**, **aggregation**, **clustering** and **filtering**. However, as previously said, in our system they are optional and each one can be activated or deactivated using the configuration file.

2.2.3.2.1 Normalization We start by normalizing candidate answers that belong to categories NUMERIC:COUNT and NUMERIC:DATE. In these cases, we attempt to diminish the variation of the answers by reducing them to a canonical representation. Since our recognizer is able to extract entities written with numeric and alphabetic characters (and both), this

[12] A gazetteer is a geographical dictionary, typically used to identify places. However, we use the term in a more broader sense to refer to a dictionary of any type of entities.

representation allows comparisons between answers: for instance, *one thousand* and *1000* are both reduced to *1000.0*.

2.2.3.2.2 Aggregation After being normalized, candidate answers are aggregated by lexical equivalence. The goal is to reduce the number of candidate answers by merging those that are lexicographically equal (insensitive case) into a single candidate answer. The score of the new answer is the sum of the scores[13] of all the answers it comprises.

For example, in the above example of retrieved passages for the question *Who painted the Birth of Venus?* (see Section 2.1), the candidate answers *Sandro Botticelli*, *Sandro Botticelli*, *Botticelli*, *Sarah Dunant* and *Alessandro di Mariano di Vanni Filipepi* are extracted with score 1.0. After this step, the (now unique) answer *Sandro Botticelli* is scored 2.0, due to the aggregation the two equal answers *Sandro Botticelli* and *Sandro Botticelli*.

2.2.3.2.3 Clustering Once equal candidate answers have been aggregated, we perform a clustering step. For that purpose, it is required the definition of a distance measure, which determines how the similarity of two candidate answers is calculated. In JUST.ASK, we have the possibility to chose from the *overlap distance* and the *Levenshtein distance* [Levenshtein, 1966] normalized to the maximum length of the two answers being compared (notice, however, that other measures can be easily integrated in JUST.ASK).

The Levenshtein distance measures the least number of edit operations to transform one string into another, while the overlap distance measures the distance between two sets (a token is an object in the set) and returns a value of 0.0 for candidate answers that are either equal or one is contained in the other (without taking into account the order of the tokens).

In either cases, the lower the distance, the similar the strings are. The chosen distance is used in conjunction with a standard single-link agglomerative clustering algorithm, which works as follows. Initially, every candidate answer starts in its own cluster. Then, at each step, the two closest clusters, up to a specified threshold distance, are merged. The distance

[13] With the exception of candidate answers that are extracted using regular expressions, all other candidate answers have a score of 1.0. Being so, in most scenarios, the score of each new answer boils down to the number of answers it aggregates.

between two clusters is considered to be the minimum of the distances between any members of the clusters, as opposed to complete-link clustering, which uses the maximum.

To illustrate the clustering algorithm at work, used in conjunction with the overlap distance with a threshold of 0.0, consider the following set of candidate answers: {*Sandro Botticelli*, *Botticelli*, *painter Sandro Botticelli*, *painter Michelangelo*}. In the first step, *Sandro Botticelli* and *Botticelli* are merged together. In the second step, *painter Sandro Botticelli* is merged with the resulting cluster from the previous step. Finally, since the minimum distance from *painter Michelangelo* to the cluster {*Sandro Botticelli*, *Botticelli*, *painter Sandro Botticelli*} is 0.5, and this value is greater than the threshold, the algorithm halts.

In addition, the representative answer of each cluster is defined as the most informative, that is, the longest answer. JUST.ASK uses a set of heuristics to choose the representative answer among the candidate answers in the cluster. First, it uses the score of the answer (recall that, if the answers have been aggregated, their score is the sum of the scores of the answers it aggregates). In case of a tie, the system uses the most informative answer. For instance, in the cluster {*Sandro Botticelli*, *Botticelli*, *painter Sandro Botticelli*}, *painter Sandro Botticelli* is selected as representative answer of this cluster. Again, in the case of a tie, JUST.ASK uses the rank and alphabetical order of the answers.

Moreover, a score is assigned to each cluster, which is simply the sum of the scores of all candidate answers within it.

2.2.3.2.4 Filtering After the clusters of candidates have been built, and in order to remove undesired answers, we apply a simple filter to our clusters. If any of the answers present in any of the clusters is contained in the original question, than the whole cluster is discarded. To understand the importance of this filter, consider the question *Which painters influenced Sandro Botticelli?*, classified as HUMAN:INDIVIDUAL. For this question, *Sandro Botticelli* and *Botticelli* are extracted as candidate answers, since both answers match the named entity type associated with the question's category (PERSON), and, due to their similarity, they are

clustered together. However, it is clear that none is the answer that is sought. Moreover, since the *Sandro Botticelli* answer appears in almost every passage, as the formulated query itself contains *Sandro Botticelli*, this will result in a very high score for it. Thus, in order to prevent this unwanted answer to be returned, the entire cluster is discarded.

2.2.3.2.5 Selection Finally, after these intermediate steps, the representative answer of the cluster with highest score is returned. Furthermore, in case two clusters happen to have the same score, the tie is settled by returning the answer of the cluster with highest search rank – i.e., the cluster whose answers were in the first results returned by the information source.

2.3 Evaluation Corpora

In this section we describe the corpora used as reference, and against which we compare the system's results.

2.3.1 GoldWebQA

The Web is constantly changing and, due to this reason, results attained by a Web QA system might differ in every run. In addition, it is probable that some questions that are plausible during a certain time period, no longer make sense some time after. For instance, the question *How old is Deng Xiaoping?*, from the Multisix corpus [Magnini et al., 2003b], makes no sense anymore, since Deng Xiaoping died in 1997. Moreover, some question/answer pairs become obsolete through time. As an example, the answer to the question *Who is the Japanese Prime Minister?* was, in 1994, *Morihiro Hosokawa* and, at the time of our experiments (2011- 2012), was *Noda Yoshihiko*. Furthermore, each correct answer can be stated in multiple forms, posing problems in a QA system evaluation if all the possible reformulations of an answer are not present in the reference. For instance, if a list of correct answers contains uniquely the answer *"March 21, 1961"* to the question *When was Lothar Matthaeus born?*, the answers *"March 21st 1961"*, *"Tuesday, March 21, 1961"*, *"in March of 1961"*, *"21-Mar-1961"*, *"21st March*

1961" and "on 21st March 1961" will be considered incorrect. Hence, we decided to build a gold corpus – the GoldWebQA – that tries to puzzle out all these problems.

The GoldWebQA consists of 200 questions, a set of snippets retrieved from the Web that contain possible answers to the questions, all the correct answers occurring in the snippets regardless of the format in which they are stated, and the category of the questions, according to Li and Roth's question type category. All the questions were plausible and valid when the snippets were retrieved (representing a snapshot of the Web).

The set of questions from the GoldWebQA is an updated and extended version of the Multisix corpus [Magnini et al., 2003b], also composed of 200 English questions and the respective correct answers, manually collected from the Los Angeles Times corpus of the year of 1994, that was initially built for the cross-language tasks at CLEF QA-2003. Questions from the GoldWebQA were obtained as follows:

1. Each question from Multisix was manually validated, to eliminate questions that no longer make sense. With this, 12 questions were removed from the corpus.

2. Each answer from Multisix was manually updated, as there were many obsolete answers.

3. The first 12 questions belonging to categories DESCRIPTION:DEFINITION and HUMAN:DEFINITION from the UIUC dataset (see Section 2.5.1.1 for details) were added to the corpus, in order to deal with the lack of questions from these categories in Multisix (non factoid-like questions).

4. The 200 questions were manually classified according to Li and Roth's question type taxonomy.

The number of questions per category and question word in the GoldWebQA corpus is presented in Tables 2.2 and 2.3, respectively.

In what concerns the retrieved snippets, they allow to truly simulate the functioning of an online QA system that resorts to a Web search engine. Moreover, besides protecting against

Category	# Questions		Category	# Questions	
DESCRIPTION:DEFINITION	6		LOCATION:CITY	8	
DESCRIPTION:DESCRIPTION	2	**9**	LOCATION:COUNTRY	4	
DESCRIPTION:MANNER	1		LOCATION:MOUNTAIN	1	**39**
ENTITY:ANIMAL	1		LOCATION:OTHER	25	
ENTITY:CREATIVE	10		LOCATION:STATE	1	
ENTITY:MEDICINE	2		NUMERIC:COUNT	24	
ENTITY:OTHER	1	**21**	NUMERIC:DATE	30	
ENTITY:PLANT	1		NUMERIC:DISTANCE	2	**63**
ENTITY:SUBSTANCE	3		NUMERIC:MONEY	2	
ENTITY:TERM	3		NUMERIC:PERCENT	1	
HUMAN:DESCRIPTION	6		NUMERIC:PERIOD	4	
HUMAN:GROUP	21	**64**	ABBREVIATION:ABBREVIATION	1	**4**
HUMAN:INDIVIDUAL	37		ABBREVIATION:EXPANSION	3	

Table 2.2: Number of questions per category in the GoldWebQA corpus.

Question word	Who(m)	When	Where	What	Which	How	Name	Other
# Questions	36	25	22	50	8	29	8	22

Table 2.3: Number of questions per question word in the GoldWebQA corpus.

the changes occurring in the Web, their use has several advantages when compared to the use of the *real* Web, namely: it reduces the latency time of the system when waiting for an external component; and, it allows benchmarking of systems that are, by nature, very difficult to evaluate, in particular in the answer extraction/selection steps.

In order to build the set of snippets plus the reference:

1. From each question of the GoldWebQA, we built keyword-based queries (as described in Section 2.2.2.2), and collected 64 snippets using the search engines Google and Yahoo.

2. We manually updated the GoldWebQA by adding variations of each correct answer that were found in the snippets to the existing correct answers.

To mention some statistics of this test collection, there is an average of 3.89 correct answers per question with standard deviation of 5.82. The top 6 questions with higher number of correct answers (from 21 to 50) start with the question word *Name*, like *Name a German philosopher*.

2.3.2 TREC-QA_2002-2007

The TREC-QA_2002-2007 corpus was built from the freely available data of the TREC QA tracks, from the years 2002 to 2007, and it is composed of questions and their answers. In particular:

- the questions expect short-size, factual answers. Given that TREC 2002 focused on factoid questions and in TRECs 2003 to 2007 the questions are annotated with their type (`factoid`, `other`, `description`, `list`), it was possible for us to extract uniquely the fact-based questions.
- the answers are those given by the competing systems and judged as correct or unsupported by the human assessors.

The reason why we refrained from using the data from all TREC QA tracks[14] in our automatic evaluation of JUST.ASK lies in the fact that, in the years 1998-2001, the exact answers to the questions are not available, rather than a text snippet that contains the answer. Moreover, from the original corpus, the following questions were discarded:

- anaphoric questions (*e.g.*: *How often does it approach the earth?* or *What is the monetary value of the prize?*), since at the present moment JUST.ASK does not deal with this linguistic phenomenon. This was important specially because after TREC 2004 the test set organized the questions into groups, each group related to a topic, and many of those test questions included an (anaphoric) reference to the topic, or to the answer or entity of a previous question in the group;
- questions that require multiple semantically distinct answers (*e.g.*: *In what city and state are Habitat for Humanity International headquarters?*);
- outdated questions, either because the question no longer makes sense (*e.g.*: *Who is Tom Cruise married to?* as at the present moment Tom Cruise is not married) or

[14] Recall that the QA track in TREC was active from 1998 to 2007.

because the answers to the question are no longer correct (*e.g.*, Sundquist is no longer the correct answer to *Who is the governor of Tennessee?*).

Therefore, in the subset of the corpus corresponding to years 2004-2007, for the cases of questions with pronominal anaphora we used a set rules to automatically filter out the questions containing the pronouns `"he"`,`"she"`,`"him"`,`"her"`,`"his"`,`"hers"`,`"their"`,`"they"`,`"it"` and `"its"`. The remaining data was manually verified and the unwanted questions were removed.

The resulting corpus contains 1440 questions, with an average of 2.34 correct answers per question. Table 2.4 shows the number of questions gathered from each year of the TREC competition that compose the TREC-QA_2002-2007 corpus.

TREC year	2002	2003	2004	2005	2006	2007	Total
# Questions	447	378	66	141	209	199	1440

Table 2.4: Number of questions in the TREC-QA_2002-2007 corpus gathered from each year of the TREC competition.

The number of questions per question word in the TREC-QA_2002-2007 corpus is shown in Table 2.5.

Question word	Who(m)	When	Where	What	Which	How	Name	Other
# Questions	121	171	84	666	23	243	1	131

Table 2.5: Number of questions per question word in the TREC-QA_2002-2007 corpus.

Table 2.6 shows examples of Q/A pairs in the TREC-QA_2002-2007 corpus.

Q/A seed pairs
(*When did the U.S. Congress approve CAFTA?*, *2005*)
(*Who is the father of the U.S. Navy?*, *John Paul Jones*)
(*Where is Merrill Lynch headquartered?*, *New York*)

Table 2.6: Some Q/A pairs in the reference corpus.

Regarding the information sources from where to extract answers, Bing is used as search engine to passage retrieval. To avoid the constant changes of the Web and of the algorithms behind the search engine, the data was retrieved and previously stored. Data was collected between September 24, 2012 and October 10, 2012.

2.3.3 On the sources of unstructured information

A characteristic of the pipelined architecture in systems of different natures is the fact that errors in a certain component are propagated through the pipeline, resulting in failures of downstream components. In particular in JUST.ASK, not being able to retrieve relevant passages/documents causes the failure of the Answer Extraction component and, therefore, of the entire system.

When evaluating JUST.ASK, we had to decide which source of unstructured information (and, thus, which search engine) to use: a closed **document-collection** or the **web**. Several (positive and negative) factors were weighted, which we summarize in the following:

Web

Positive – it is highly redundant: answers are stated in multiple forms, in multiple documents;

Positive – nowadays, web search engines are a focus of intense research and subject of constant changes and improvements;

Negative – it is not stable: changes occur over time with the introduction/indexation of new information. Thus, experiments are not replicable;

Negative – many web search APIs are becoming deprecated and even closed (for example, the Google Web Search API has been deprecated as of November 1, 2010.) In other cases, a payment is required to access the search results after a certain limit of queries (Bing starts charging after a limit of 5000 queries/month). Also, some search engines do not allow the local storage of the results for more than a certain period of time, which makes it hard to rerun experiments.

Document Collection

Positive – it is stable over time, does not change. Therefore, the approaches and results can be replicable;

Negative – there is not enough redundancy (when compared to the Web);

Negative – it has to be locally indexed, requiring some fine tuning in order to achieve effective results;

Negative – the availability of corpora often depends on license agreements. In particular, we do not have free access to a part of the collections used in the TREC competition of the year 2002 (the AQUAINT Corpus of English News Text).

In the evaluation of JUST.ASK, we adopted an hybrid between the Web and a document collection: we used the web as source of the information, but stored locally the search results associated with each query. Therefore, we take advantage of the redundancy of the web and of the recent advances done to the search engine, while not compromising future tests of our approach.

Note that, the fact that we decided to use the web as source of information also influenced our choice of which retrieval unit to use: passages (*i.e.*, the small summary retrieved by the search engine for a given query) or webpages/documents. Since the web is composed of pages that follow a hight variety of formats, we opted to use passages instead of the content of the webpage/document so we did not have to deal with the problem of webpage pre-processing (*e.g.*, removal of advertisements, navigation bars,) and content extraction, which represents in itself a body of research [Baroni et al., 2008].

2.4 Evaluation Measures

To assess the performance of JUST.ASK and its components, we use several measures that have been proposed and extensively reported in the literature, namely for the evaluation of QA systems [Voorhees, 2003a]. We also include a metric (*coverage*) that indicates the proportion of answered (correct or wrong) questions in the total questions. In the evaluation of JUST.ASK, the following measures are used:

Accuracy, defined as the proportion of questions correctly answered:

$$Accuracy = \frac{\#Correctly\ answered\ questions}{\#Questions\ in\ the\ test\ corpus}\,. \tag{2.1}$$

Recall, defined in QA in a non-standard way, corresponding to the definition of accuracy [Artstein, 2011]. In this thesis, we use the term recall whenever we compute the F-measure; otherwise, we use accuracy.

Precision, defined as the proportion of questions answered correctly in the total of questions answered:

$$Precision = \frac{\#Correctly\ answered\ questions}{\#Answered\ questions}\,. \tag{2.2}$$

***F*-measure**, used to combine the above two measures into a single metric, and it is defined as a weighted harmonic mean of precision and recall:

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall}\,, \tag{2.3}$$

where the parameter β is used to adjust the weights that are given to each measure. A value of $\beta > 1$ favours recall, whilst a value of $\beta < 1$ favors precision. When precision and recall are equally weighted (i.e., $\beta = 1$), the measure usually goes by the name of F_1.

Mean Reciprocal Rank (MRR), used to evaluate systems that return ranked lists of items to a query. The reciprocal **rank** of an individual query is defined to be the multiplicative inverse of the rank of the first relevant item, or zero if no relevant items are returned. Thus, the mean reciprocal rank is the average of the reciprocal rank of every query in a test corpus. More formally, for a test collection of N queries, the MRR is given by:

$$MRR = \frac{1}{N}\sum_{i=1}^{N}\frac{1}{rank_i}\,. \tag{2.4}$$

Coverage, defined as the proportion of questions answered in the total of questions:

$$Coverage = \frac{\#Answered\ questions}{\#Questions\ in\ the\ test\ corpus}\,. \tag{2.5}$$

2.5 Baseline Evaluation

2.5.1 Results – JustAsk@GoldWebQA

2.5.1.1 Question processing results

In what concerns the Question Processing component, the evaluation is focused on the question classifier, due to its vital importance for the QA task.

The classifier of JUST.ASK was tested using the GoldWebQA corpus where it was able to correctly classify 185 questions, resulting in an accuracy of 92.5% for the fine-grained categories. The 15 questions that were not correctly classified belonged to categories DESCRIPTION (3), ENTITY (4), HUMAN (4), NUMERIC (1) and LOCATION (3). From the erroneously classified questions that should have been classified with the coarse-grained category HUMAN, all belonged to the fine-grained category GROUP (for instance, the question *Which fast-food chain cut the price of the Big Mac hamburger* was classified as ENTITY:FOOD, instead of HUMAN:GROUP). This is not a surprise as this type of questions are known to be particularly difficult to classify.

Moreover, for benchmarking purposes, we have previously carried out an intrinsic evaluation of JUST.ASK question classifier [Silva et al., 2011] on the UIUC corpus, the publicly available data set of the Cognitive Computing Group at University of Illinois at Urbana-Champaign[15], which consists of a training corpus of nearly 5500 questions and a test corpus with 500 questions, each question labeled according to Li and Roth's taxonomy for question

[15] `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`.

classification. The question classifier of JUST.ASK uses the `LIBSVM` [Chang and Lin, 2001] implementation of a SVM classifier with a linear kernel, trained with all 5500 questions from the referred training corpus, using the one-versus-all multi-class strategy.

	Granularity	
Work	Coarse	Fine
JUST.ASK	**95.0%**	**90.4%**
[Li and Roth, 2002]	91.0%	84.2%
[Zhang and Lee, 2003]	90.0%	80.2%
[Krishnan et al., 2005]	93.4%	86.2%
[Li and Roth, 2006]	92.5%	85.0%
[Blunsom et al., 2006]	91.8%	86.6%
[Moschitti et al., 2007]	91.8%	-
[Pan et al., 2008]	94.0%	-
[Huang et al., 2008]	93.6%	89.2%

Table 2.7: Comparison of accuracy results attained by JUST.ASK's Question Classifier, against other results reported in the literature that use the same train and test datasets.

Since we made use of the same corpora, either for training and testing, we can compare JUST.ASK question classifier performance with other works. The attained results are summarized in Table 2.7, and discussed in detail in Silva et al. [2011], where we show that by training a machine learning classifier with unigrams and the information generated by our rule-based classifier, higher accuracy is achieved for coarse- and fine-grained categories than the ones mentioned in state of the art literature.

As mentioned earlier, depending on the category of the posed question, JUST.ASK uses distinct information sources in the Passage Retrieval component and adopt different strategies in the Answer Extraction component. Therefore, in the following we split the evaluation into two major subsections: the first is dedicated the results achieved for non factoid-like questions (belonging to categories DESCRIPTION:DEFINITION and HUMAN:DESCRIPTION); the second is focused on the results achieved for factoid-like questions.

2.5.1.2 Non factoid-like questions

Regarding the non factoid-like questions, JUST.ASK correctly answered 9 of the 12 questions categorized as DESCRIPTION:DEFINITION or HUMAN:DESCRIPTION.

The system was not able to find the correct answer in DBPedia to two DESCRIPTION:DEFINITION questions (*What are liver enzymes?* and *What is the nature of learning?*) and to one HUMAN:DESCRIPTION question (*Who is Coronado?*). In the formers, JUST.ASK returned a wrong answer; in the latter, JUST.ASK did not return any answer.

2.5.1.3 Factoid-like questions

2.5.1.3.1 Passage retrieval results Regarding the Passage Retrieval component, the goal was to study the influence of different settings to the retrieval of passages. Specifically, we wanted to understand if there exists any significant difference when using different search engines and if there exists any significant difference when dealing with a different number of retrieved passages.

Being so, in several runs of JUST.ASK, we varied different parameters in the Passage Retrieval component, namely:

- the used search engine: either Google or Yahoo[16].
- the number of passages retrieved from the search engine: 8, 32 and 64.

With this, we could analyze the impact of each parameter in the performance of the Passage Retrieval component, particularly in the number of questions for which the retrieved passages contain at least one correct answer. For simplicity reasons, we will denote these passages as *positive passages*.

Table 2.8 present, for different amounts of retrieved passages, the number of factoid-like questions for which there is at least one positive passage (# $\text{Questions}_{1+PosPassages}$), the mean reciprocal rank of the first positive passage for all factoid-like questions (MRR_{AllQ})[17] and the mean reciprocal rank of the first positive passage only for the factoid-like questions for which at least one positive passage exists ($\text{MRR}_{QPosPassages}$).

[16] At the time of our experiments, neither of the API's was deprecated
[17] Recall that there are 188 factoid-like questions in the GoldWebQA corpus.

Google			
	# Questions$_{1+PosPassages}$	MRR$_{AllQ}$	MRR$_{QPosPassages}$
8	129 (68.6%)	0.43	0.62
32	152 (80.9%)	0.44	0.54
64	158 (84.0%)	0.44	0.52
Yahoo			
	# Questions$_{1+PosPassages}$	MRR$_{AllQ}$	MRR$_{QPosPassages}$
8	117 (62.2%)	0.34	0.55
32	146 (77.6%)	0.36	0.46
64	153 (81.4%)	0.36	0.44

Table 2.8: Passage retrieval intrinsic evaluation, while varying the number of passages retrieved from the web search engine.

The first consideration after the analysis of the table is that the passage retrieval results when using Google surpass the results attained when we use Yahoo.

Regardless of the search engine, the number of questions for which there is at least one positive passage increases as the number of retrieved passages also increase, but this amount does not pass the barrier of 84% (in the total number of factoid-like questions). Also, if all questions are considered, the increase in the number of passages retrieved by the search engine leads to an increase of the MRR of the first positive passage, even if always lower than 0.5. However, if only the questions with positive passages are considered, the MRR values decrease, indicating that, in average, the first positive passage is encountered in the tail of the set of passages. Thus, the results attained suggest that, for some cases, the first correct answer that solves a question does not appear in the top first results retrieved by search engines.

We have also inspected the questions for which neither of the search engines retrieved a positive passage. An example of such occurs, for instance, in the question *Which Elvis Presley's cover did Gilmore sing?*, where none of retrieved passages contain any of its correct answers. From the 30 questions with no retrieved positive passages when using Google and the 35 questions with no retrieved positive passages when using Yahoo, 19 overlap.

The results achieved lead us to two main considerations. On the one hand, the usage of both search engines in parallel could boost the results of the accuracy of the Passage

Retrieval component to the mark of 90% (169 positive passages retrieved for 188 queries). Indeed, it agrees with the principle behind data redundancy in which "more is better", also supported by other works in the literature, like [Lin, 2007]. On the other hand, the queries to 11 different questions could not retrieve any positive passage, regardless of the engine in use, which suggests an open research line when it comes to the understanding of questions and their translation into queries.

2.5.1.3.2 Answer extraction results The evaluation of the Answer Extraction component of JUST.ASK is important to understand if the system successfully extracts and selects the correct final answer.

Here we particularly analyze its sub-components dedicated to the tasks of candidate answer extraction and final answer selection, respectively. We use the setting in which the Passage Retrieval component of JUST.ASK yielded the best results, that is, when 64 passages were retrieved from Google. Moreover, to calculate the distance between candidate answers we used the normalized Levenshtein distance, with a threshold of 0.2 (empirically set).

Table 2.9 presents the number of questions for which a certain amount of answers was extracted. For those questions, the table shows the amount in which the candidate answer extraction sub-component was successful, and the amount in which the final answer selection sub-component was successful. We consider the candidate answer extraction stage as being successful when it extracts at least one correct answer. By the same token, the final answer selection stage is successful when a correct answer is selected to be in the list of 3 answers that the system consider as final.

Several considerations can be made from the analysis of the table. When the number of extracted candidate answers is low (between 1 and 10), if it happens to contain the correct answer (that is, the candidate answer extraction stage is successful), then the answer selection stage chooses it as final answer. When the number is higher, some questions for which a correct answer exists in the passages are filtered out in both sub-components of the Answer Extraction component. However, we can not discriminate any tendency of results to improve

# Extracted Answers	Google – 64 passages				
	Questions (#)	CAE success	FAS success	CAE accuracy	FAS accuracy
0	10	0	–	0.00%	–
1 to 10	12	4	4	33.33%	100.00%
11 to 20	10	7	5	70.00%	71.43%
21 to 40	23	16	12	69.57%	75.00%
41 to 60	40	34	25	85.00%	73.52%
61 to 100	32	29	26	90.63%	89.66%
> 100	31	27	21	87.10%	77.78%
All	158	117	93	74.05%	79.49%

Table 2.9: Answer extraction intrinsic evaluation, using 64 passages from the search engine Google. The results achieved in the candidate answer extraction (CAE) and final answer selection (FAS) stages of Just.Ask are shown.

or deteriorate with the increase of the number of extracted answers.

For comparison purposes, Table 2.10 shows the results achieved by the Answer Extraction component when 8 and 32 passages from Google are used.

As expected, the number of candidate answers per question decreases when the number of passages available also diminish. Achieved results show a compromise between the number of passages from where to extract candidate answers and the accuracy of the Answer Extraction component. Indeed, if the number of passages is too low, there is the possibility that the passages do not contain true positives (correct answers) in sufficient amount than can lead to their selection as the final answer. If the number of passages is too high, there is a chance that the system will extract too many false positives (wrong answers), leading to mistakes in the selection of the final answer.

Regarding the influence of the decrease of the number of passages on the performance of each of the stages, there is an increase of 2% on the accuracy of the candidate answer extraction stage (from 72.09% when using 64 passages to 74.05% when using 8), and a decrease of 5% on the accuracy of the answer selection stage (from 84.95% when using 64 passages to 79.79% when using 8).

In Table 2.11 we present the results of intrinsic evaluation of the Answer Extraction component of Just.Ask according to the question category. We show the total number of

# Extracted Answers	Google – 8 passages Questions (#)	CAE success	FAS success	CAE accuracy	FAS accuracy
0	16	0	–	0.00%	–
1 to 10	73	59	51	80.82%	86.44%
11 to 20	34	29	23	85.29%	79.31%
21 to 40	5	4	4	80.00%	100.00%
41 to 60	1	1	1	100.00%	100.00%
61 to 100	0	–	–	–	–
> 100	0	–	–	–	–
All	129	93	79	72.09%	84.95%
# Extracted Answers	Google – 32 passages Questions (#)	CAE success	FAS success	CAE acc	FAS acc
0	12	0	–	0.00%	–
1 to 10	18	8	8	44.44%	100.00%
11 to 20	21	14	10	66.66%	71.43%
21 to 40	57	52	41	91.23%	78.85%
41 to 60	26	22	18	84.62%	81.82%
61 to 100	14	11	10	78.57%	90.91%
> 100	4	4	3	100.00%	75.00%
All	152	111	90	73.03%	81.08%

Table 2.10: Answer extraction intrinsic evaluation, using Google and two different amounts of retrieved passages. The results achieved in the candidate answer extraction (CAE) and final answer selection (FAS) stages of JUST.ASK are shown.

questions existing for each category and, for the different amounts of retrieved passages from Google, the number of questions that resulted in the extraction of at least one candidate answer.

This table shows the categories that had higher impact on the results achieved by the candidate answer extraction and final answer selection stages of JUST.ASK. For instance, when it comes to the coarse-grained category ENTITY, the system only extracts a correct answer for 5 questions in the 21 possible. We notice also that increasing the number of passages does not always lead to better results. For example, for the category ENTITY, the best absolute results are achieved when only 8 passages are used. For categories HUMAN and LOCATION, there is a drop in the performance of the Answer Extraction component (measured by the number of questions that returned a correct answer divided by the number of questions for which a positive passage exists).

		Google – 64 passages				
	Total	Questions (#)	CAE success	FAS success	CAE accuracy	FAS accuracy
ABB	4	4	3	3	75.00%	100.00%
DES	3	2	0	–	0.00%	–
ENT	21	17	5	2	21.41%	40.00%
HUM	58	51	42	35	82.35%	83.33%
LOC	39	35	29	24	82.86%	82.76%
NUM	63	49	38	29	77.55%	76.32%
	188	158	117	93	74.05%	79.49%
		Google – 32 passages				
ABB	4	3	2	2	66.66%	100.00%
DES	3	2	0	–	0.00%	–
ENT	21	16	3	2	18.75%	66.66%
HUM	58	50	41	33	82.00%	80.48%
LOC	39	34	28	25	82.35%	89.29%
NUM	63	47	37	28	78.72%	75.68%
	188	152	111	90	73.03%	81.08%
		Google – 8 passages				
ABB	4	3	1	1	33.33%	100.00%
DES	3	2	0	–	0.00%	–
ENT	21	15	3	3	20.00%	100.00%
HUM	58	42	35	32	83.33%	91.43%
LOC	39	27	24	21	88.89%	87.50%
NUM	63	40	30	22	75.00%	73.33%
	188	129	93	79	72.09%	84.95%

Table 2.11: Answer extraction intrinsic evaluation, for different amounts of retrieved passages, according to the question category. The results achieved in the candidate answer extraction (CAE) and final answer selection (FAS) stages of JUST.ASK are shown.

The achieved results clearly evince the categories in which more efforts should be invested. For instance, the main improvement for the category ENTITY should be in the extraction of candidate answers, while for categories HUMAN and NUMERIC it should rather be in the selection of the final answer.

2.5.1.3.3 Overall results of the factoid-like questions The best result of JUST.ASK in the GoldWebQA corpus was achieved when we used the first 64 passages from Google. Table 2.12 presents the system's overall results. Since JUST.ASK returns a list with 3 possible final answers to a given question, the precision, recall, F_1-measure, and MRR are measured and shown for the top 3 answers returned by the system. In this table, we also present the

results for the top one answer returned (we denote it as accuracy@1).

# Questions		Correct	Wrong	Unanswered
188		93	77	18
Precision	Recall	F_1-measure	MRR	Accuracy@1
54.7%	49.5%	0.52	0.40	30.0%

Table 2.12: Best results achieved by JUST.ASK in the evaluation using the GoldWebQA corpus.

In a total of 188 questions, JUST.ASK gave answers to 170. From these, 93 were correct and 77 wrong. Being so, the system attained a recall of 49.5%. Moreover, the results in accuracy@1 show that, from the 93 correctly answered questions, the system was able to push the correct answer to the first position of the list in 60 questions (this result is not shown in the table).

Detailed results Table 2.13 details the achieved results according to the question category. The result that first pops out is that for the category DESCRIPTION the system had an accuracy of 0.0%. The second worst result is that of category ENTITY, for which the system failed to answer 13 questions and skipped 6, from a total of 21. For instance, the system did not gave any answer to the question *What is the Chicken Boy sculpture made of?* and gave the wrong answer to *What flower was named after the first lady Hillary Rodham Clinton?*: *"orchid"* instead of *"tulip"*.

Category	Correct	Incorrect	Unanswered	Accuracy
ABBREVIATION	3	1	0	75.00%
DESCRIPTION	0	0	3	0.00%
ENTITY	2	13	6	9.50%
HUMAN	35	20	3	60.34%
LOCATION	24	15	0	61.54%
NUMERIC	29	28	6	46.03%
Total	93	77	18	49.47%

Table 2.13: JUST.ASK results according to the different question categories.

In Table 2.14 we show the system's results depending on the question word. Whereas JUST.ASK demonstrates a good performance in questions that typically involve the name or definition of a person (question word "Who(m)"), the worst results happen for questions that start with "How".

Question word	Correct	Incorrect	Unanswered	Accuracy
Who(m)	19	11	0	63.33%
When	18	7	0	72.00%
Where	12	10	0	54.55%
What	17	15	12	38.64%
Which	4	3	1	50.00%
How	8	16	5	27.59%
Name	6	2	0	75.00%
Other	9	13	0	40.91%
Total	93	77	18	49.47%

Table 2.14: Results according to the different question words.

Impact of the components As previously said, JUST.ASK follows the typical pipelined architecture. Thus, it is relevant to understand which components are failing, since this failure will reflect on the following components. Being so, we consider that the question processing fails when it does not attribute the correct category to a question, the passage retrieval fails if it does not retrieve at least one positive passage for a question, and the answer extraction fails when it does not extract and selects the correct answer for a question.

For each of the 95 wrong and unanswered questions, we analyzed which component is the first one failing. The results on Table 2.15 show the impact on the final answer of each component independently, and do not account for the failures in cascade that occur in the system due to its pipelined architecture.

Component	# Questions
Question Processing	15
Passage Retrieval	24
Answer Extraction	56
	95

Table 2.15: Number of times each component is the first to fail, avoiding the system to return the correct answer to 95 (wrong or unanswered) questions.

From the table, we can see that:

- in a total of 188 factoid-like questions, the Question Processing component succeeded in 173 (92.02% of the questions) and failed in 15;
- in a total of 173 correctly classified questions, the Passage Retrieval succeeded in 158

(84.04%) and failed in 24; and,

- in a total of 158 correctly classified questions for which at least one positive passage was retrieved, the Answer Extraction succeeded in 93 (49.47%) and failed in 56.

There is still work to be done in JUST.ASK components: the Question Processing was properly tuned, reflecting in a small number of failures due to this component; when it comes to the Passage Retrieval component, we believe that one way to improve our results may imply the recourse to query expansion; regarding the Answer Extraction, it requires deeper improvements, and probably new strategies, as most of our failures originate in this component.

2.5.2 Results – Just.Ask@TREC-QA_2002-2007

In the evaluation of JUST.ASK using the TREC-QA_2002-2007 corpus, the following configuration was used:

In the Passage Retrieval component, we used all the search results returned for each query (JUST.ASK asked 100 search results to the engine Bing).

In the Answer Extraction component, in the clustering step, the distance between candidate answers is calculated using the normalized Levenshtein distance with a threshold of 0.2.

Moreover, as the TREC-QA_2002-2007 corpus is composed uniquely by factoid-like questions, we disabled the Passage Retrieval and Answer Extraction strategies for non factoid-like questions (that is, those based on the Wikipedia/DBPedia).

JUST.ASK returns a list of possible final answers, which are the representative answers of the cluster with highest scores, and, in the typical evaluation, each answer is evaluated against the reference. The downside of this approach is that if there is a slight lexical difference between the reference and the system's answer the latter is considered wrong. For example, if the reference answer is *August 1987* and the system picks the answer *Aug. 1987* as final from

the cluster of candidates (*Aug. 1987*, *08-1987* and *August 1987*), it is evaluated as wrong. This is particularly necessary since the TREC-QA_2002-2007 corpus lacks variations of the correct answers that allow us to precisely identify if an answer is correct or not (which can be particularly important in the case of NUMERIC answers), a situation that did not occur in the GoldWebQA corpus, as it contains all possible variations of the answers contained in the information sources. Being so, to circumvent this issue, when evaluating the system's results using the TREC-QA_2002-2007 corpus, we consider every answer in the cluster, rather than only its representative.

2.5.2.1 Question processing results

Table 2.16 indicates the number of questions per category in the TREC-QA_2002-2007 corpus, according to JUST.ASK's question classifier.

The TREC-QA_2002-2007 corpus does not contain the reference categories for each question. Therefore, we picked a random sample of 100 classified questions which we manually evaluated. Results showed that the classifier correctly classified 87% of the questions.

One point that deserves particular attention is the existence of 42 questions classified as DESCRIPTION:DEFINITION and 1 classified as HUMAN:DESCRIPTION. Recalling that all questions expect short-sized fact-based answers (as mentioned when the characteristics of the TREC-QA_2002-2007 corpus were presented), this results in the fact that at least 43 questions are wrong (or unanswered) due to errors in the question processing step.

2.5.2.2 Passage retrieval results

For the total 1440 questions, the passage retrieval component retrieved at least one passage for 1437, being 99.4 the average number of retrieved passages for each question. However, the average number of positive passages retrieved per question is much lower (15.8). Moreover, in only 1216 of these questions at least one positive passage was retrieved, meaning that 224 questions will not be answered due to failures in the passage retrieval component. Table 2.17

Category	# Questions		Category	# Questions	
DESCRIPTION:DEFINITION	42	**100**	HUMAN:DESCRIPTION	1	**298**
DESCRIPTION:DESCRIPTION	14		HUMAN:GROUP	78	
DESCRIPTION:MANNER	37		HUMAN:INDIVIDUAL	213	
DESCRIPTION:REASON	7		HUMAN:TITLE	6	
ENTITY:ANIMAL	24	**204**	LOCATION:CITY	56	**293**
ENTITY:BODY	2		LOCATION:COUNTRY	57	
ENTITY:COLOR	6		LOCATION:MOUNTAIN	4	
ENTITY:CREATIVE	39		LOCATION:OTHER	157	
ENTITY:EVENT	6		LOCATION:STATE	19	
ENTITY:CURRENCY	1		NUMERIC:CODE	2	**524**
ENTITY:FOOD	10		NUMERIC:COUNT	124	
ENTITY:INSTRUMENT	4		NUMERIC:DATE	271	
ENTITY:LANGUAGE	4		NUMERIC:DISTANCE	45	
ENTITY:MEDICINE	5		NUMERIC:MONEY	12	
ENTITY:OTHER	43		NUMERIC:OTHER	12	
ENTITY:PLANT	1		NUMERIC:PERCENT	11	
ENTITY:RELIGION	1		NUMERIC:PERIOD	25	
ENTITY:SPORT	3		NUMERIC:SIZE	9	
ENTITY:SUBSTANCE	11		NUMERIC:SPEED	8	
ENTITY:SYMBOL	2		NUMERIC:TEMPERATURE	4	
ENTITY:TECHNIQUE	1		NUMERIC:WEIGHT	1	
ENTITY:TERM	32		ABBREVIATION:ABBREVIATION	2	**21**
ENTITY:VEHICLE	6		ABBREVIATION:EXPANSION	19	
ENTITY:WORD	3				

Table 2.16: Number of questions per category in the TREC-QA_2002-2007 corpus.

summarizes those results and includes the mean reciprocal rank of the first positive passage for all questions and for the questions for which at least one positive passage exists.

Bing			
#Questions$_{1+Passages}$	# Questions$_{1+PosPassages}$	MRR$_{AllQ}$	MRR$_{QPosPassages}$
1436 (99.7%)	1215 (84.4%)	0.29	0.35

Table 2.17: Passage retrieval intrinsic evaluation.

2.5.2.3 Answer extraction results

For the 1215 questions with positive passages, JUST.ASK could extract and select the correct answer to 791 and 678 questions, respectively. The results of the answer extraction component are displayed in Table 2.18, where it can also be seen that, in most questions, JUST.ASK extracts a large amount of candidates and the accuracy in the selection of the final answer

decreases when the number of candidate answers is higher. In 186 questions for which there was at least one positive passage, the system is not able to extract any correct answer.

	Bing – 100 passages				
# Extracted Answers	Questions (#)	CAE success	FAS success	CAE accuracy	FAS accuracy
0	186	0	–	0.00%	–
1 to 10	51	18	18	35.29%	100.00%
11 to 20	68	33	32	48.53%	96.97%
21 to 40	157	106	96	67.52%	90.57%
41 to 60	148	117	106	79.05%	90.60%
61 to 100	231	186	152	80.50%	81.72%
> 100	374	331	274	88.50%	82.78%
All	1215	791	678	65.10%	85.71%

Table 2.18: Answer extraction intrinsic evaluation, using 100 passages from the search engine Bing.

2.5.2.4 Overall results

Table 2.19 presents the system's overall results for the TREC-QA_2002-2007 corpus. Again, the precision, recall, F_1-measure, and MRR are measured and shown for the top 3 answers (clusters) returned by the system. In this table, we also present the results for the top one answer returned (denoted as accuracy@1).

# Questions		Correct	Wrong	Unanswered
1440		678	511	251
Precision	**Recall**	**F_1-measure**	**MRR**	**Accuracy@1**
56.7%	47.1%	52.6%	0.51	36.3%

Table 2.19: Results achieved by JUST.ASK in the evaluation using the TREC-QA_2002-2007 corpus.

In a total of 1440 questions, JUST.ASK returned answers to nearly 83%. From these, 678 were considered correct and 511 wrong. Therefore, the system's recall was of 47.1%. Moreover, in 522 questions, the correct answer was the one in the highest position of the list of returned answers.

Detailed results Table 2.20 shows the system's results according to the question categories. The first thing to be mentioned is that the system did not try to answer any of the DESCRIP-

TION questions (recall that the strategies for non factoid-like questions were disabled). On the contrary, it answered all the LOCATION questions, and it was successful in nearly 70% of them.

Category	Correct	Incorrect	Unanswered	Accuracy
ABBREVIATION	14	2	5	66.67%
DESCRIPTION	0	0	100	0.00%
ENTITY	29	76	99	14.22%
HUMAN	147	145	6	49.33%
LOCATION	200	93	0	68.26%
NUMERIC	288	195	41	54.96%
Total	678	511	251	47.08%

Table 2.20: JUST.ASK results according to the different question categories.

In Table 2.21 the system's results according to the question word are presented. The lowest performance (ignoring the *Name* question) occurs in the *How* and *What* questions: regarding the former, JUST.ASK only answers around 75% (183 in 243) and, from these, less than 44% are correct; regarding the latter, JUST.ASK leaves around 25% of the questions unanswered (174 in 666) and from the remaining, it correctly answers only 52.24% (257 in 492). We observed that all questions marked as DEFINITION start with the question word *What*.

Question word	Correct	Incorrect	Unanswered	Accuracy
Who(m)	71	49	1	58.68%
When	135	35	1	78.95%
Where	56	28	0	66.67%
What	257	235	174	38.53%
Which	17	5	1	73.91%
How	79	104	60	32.51%
Name	0	1	0	0.00%
Other	63	55	14	47.73%
Total	678	511	251	47.08%

Table 2.21: JUST.ASK results according to the different question words in the TREC-QA_2002-2007 corpus

2.6 Discussion

2.6.1 Comparison with other systems

In the evaluation with the GoldWebQA corpus, JUST.ASK achieved an accuracy@1 of 31.9%, with 60 correct questions in a total of 188. If we consider the top three answers returned by the system, the accuracy increases to almost 50%, with 93 correctly answered questions (see Table 2.12).

A direct comparison with the other systems evaluated with the corpus in which the GoldWebQA is based (the Multisix corpus) is not possible. This is due to the fact that the Multisix corpus was used mainly in bilingual QA,[18] while in monolingual tasks the English language was not used. Nevertheless, it is worth mentioning that, in the monolingual task using other languages (namely, Spanish, Italian and Dutch), the competing systems returned between 77 and 101 correct questions [Magnini et al., 2003a],[19] in a setting where a maximum of 3 answers were allowed per question and unsupported answers also accounted as correct. Therefore, the values of accuracy@3 ranged between 38.5% and 50.05%. Regarding the MRR, it ranged between 0.317 and 0.442; JUST.ASK achieved 0.40. In the monolingual task, answers were to be found in a document collection of the same language as the question (recall that JUST.ASK searched for the answers in the Web), while in the bilingual task, the systems are supposed to find the answer in English, which implies a task of translation in the QA pipeline. This resulted in a decrease of the overall systems performance that ranged between an accuracy of 45% (90 correct answers, MRR of 0,40, in Italian → English) and 11.5% (23 correct answers, MRR of 0,115, in French → English).

In the evaluation with the TREC-QA_2002-2007 corpus, JUST.ASK attained an accuracy of 47.1% in 1440 questions, with 678 correct questions within the top 3 returned answers. In 522 questions, the system was able to push the correct answer to the first position of the list

[18] In bilingual QA, systems receive a question in a source language and are expected to return an answer in a different target language. In the CLEF 2003 evaluation, in the bilingual task, the answer was to be searched in an English document collection.

[19] We report results of the *lenient* evaluation, where the unsupported responses are also considered correct.

of returned answers, resulting in an accuracy@1 of 36.3% (see Table 2.19).

Again, a direct comparison between JUST.ASK and the other systems is not possible, mostly due to the fact that we are not using the same questions nor the same information sources. Although the test questions used in TREC are freely available, as we mention in Section 2.3.3, we do not have license to access all the information sources used in the competitions. Moreover, another reason that does not allow us to directly compare the results lies on the fact that the answers at TREC are evaluated manually, by human accessors, and ours was made automatically, by matching JUST.ASK answers with the correct answers that the systems competing at TREC returned. Therefore, there might be correctly extracted answers by JUST.ASK that were not in the reference corpus and, thus, not accounted as correct. Finally, an answer is correct in the TREC evaluation if it contains the passage that supports it; in our evaluation, we do not required the presence of a supporting snippet to consider an answer as correct.

Despite these contrarieties, for the purpose of this comparison, we split the questions in the TREC-QA_2002-2007 corpus according to the different years (some statistics about the division of the corpus will be given in the next chapter, in Table 3.3) and compare JUST.ASK's accuracy@1 results with the results achieved on the best runs of the top systems at TRECs 2002-3-4-5-6-7 for the factoid component.

In 2002, JUST.ASK ranks 4^{th} with an accuracy@1 of 44.5% in a year where the best result was of 83.0% and the worse was of 21.0% [Voorhees, 2002]. In the year of 2003, competing systems results ranged from 14.5% to 70.% [Voorhees, 2003b] and our system ranks 5^{th} with an accuracy@1 of 32.0%. The question topics were introduced in 2004, a year where accuracy results ranged between 21.3% and 77.0% [Voorhees, 2004] and JUST.ASK is positioned at number 4, with an accuracy@1 of 42.42%. In 2005, JUST.ASK accuracy@1 was of 34%, positioned 3^{rd} in a list where the best result was of 71.3% and the worst of 21.5% [Voorhees and Dang, 2005]. In 2006, our system's accuracy@1 was of 32.1%; the best and worst run attained 57.8% and 21.3% [Dang et al., 2006], respectively, our system lays at number 5 in the list. Finally, in 2007, JUST.ASK is positioned in 3^{rd}, with an accuracy@1 of 29.6%; in this

year the best result was of 70.6% and the worst of 20.6% [Dang et al., 2007].

2.6.2 Strengths and weaknesses of Just.Ask

The conducted evaluations allowed to identify the strong and weak points of the system, stated in the following:

- The question classifier of Just.Ask attains state-of-the-art results when evaluated against the UIUC corpus. When using the GoldWebQA corpus, it reached 92.5% of accuracy and in the TREC-QA_2002-2007 corpus the accuracy was of 87%. This step is surely one of the basis of the system, since no question incorrectly classified is correctly answered.

- Regarding the Passage Retrieval component, results suggest that the increase on the number of passages does not always lead to better overall performance. Indeed, for some question categories, increasing from 8 to 32 passages compromises the relative accuracy of the Answer Extraction component. This is mostly due to the fact that many more wrong answers are involved in the answer selection stage.

- Another important conclusion is that a considerable number of questions resulted in the retrieval of no passages with a correct answer (around 15% in the evaluation with both the GoldWebQA and the TREC-QA_2002-2007 corpora, for 64 and 100 maximum passages retrieved respectively). Query expansion is, thus, a desirable research direction.

- The Answer Extraction component also requires improvements. In the evaluation using the GoldWebQA corpus it was responsible for 56 of the 95 failed questions (nearly 60%), while in the evaluation using the TREC-QA_2002-2007 corpus, 537 questions were not answered due to failures in this component (more than 37% of the total questions). One of the weak points of Just.Ask lies exactly in the Answer Extraction component, since even when Just.Ask extracts the correct answer from the information sources, often it is not able to select it.

- In overall terms, considering the different categories, the questions that belong to category ENTITY as well as questions that start with *How* need improvements.

2.6.3 On the evaluation of question-answering systems

One of the most challenging tasks in QA has to do with the systems' evaluation and comparison with others, due to two main reasons:

- Lack of evaluation resources: existing test questions focus on different time frames and require different information sources. In fact, and besides some specific sub-tasks of QA that are already evaluated as a separate component (like the question classification), there is still a lack of resources (questions, information sources and, more important, the correct answers) in enough amount to allow a large-scale evaluation and a fair comparison of systems. This issue is even more pernicious when the targeted language is not English. While it is certain that the competitions promoted between QA systems, like those at TREC and CLEF, greatly boosted the availability of resources, it is also true that many of those resources (specially, the information sources) were restricted to the participating systems and are not freely available.

- Need for manual evaluation: the evaluation of QA systems often requires the existence of a human judger that explicitly validates the system's answers, making it an effortful and time-consuming task. A pure automatic evaluation is typically avoided, and is often complemented with human supervision. Moreover, it is deemed imprecise if the judging mechanism does not account for the lexical and semantic variations of the correct answer to a question (for example, the answers *July 20, 1969* and 07/20/1969 to the question *What day did Neil Armstrong land on the moon?* are both correct and *committed suicide* and *killed himself* correctly answer *How did Adolf Hitler die?*).

In this thesis, two overall evaluations will be presented, using two different corpora: the GoldWebQA and the TREC-QA_2002-2007 corpora. In the former, given that we manually identified all the correct answers that the system could extract from the information sources,

we were able to have an exact perception of how the system behaves: when it is successful and when it fails. In the TREC-QA_2002-2007 corpora, a much larger dataset of questions, given that we did not have all the correct answers to the questions, we can only have an approximate idea of how the system is behaving, knowing that some questions evaluated as wrong are in fact true (just our evaluation mechanism was not 100% exact).

2.7 Summary

In this section we described JUST.ASK, the open domain QA system that will be the focus of this thesis. JUST.ASK follows the conventional pipelined architecture composed of three main components: Question Processing, Passage Retrieval and Answer Extraction. We detailed each of the components, with special attention in the answering phase in JUST.ASK. We presented two test collections built to evaluate JUST.ASK (the GoldWebQA and the TREC-QA_2002-2007 corpora), the performance metrics and reported the detailed evaluation of the system, which allowed us to identify the strong and weak points of JUST.ASK. In particular, the Answer Extraction component of JUST.ASK deserves special attention since it is the root of many of the system's wrong answers.

Given the importance of the Answer Extraction component in the overall performance of a QA system, new strategies to candidate answer extraction and final answer selection are to be devised. In the following chapters, we will address these challenges.

3 Candidate Answer Extraction based on Learned Patterns

The task of candidate answer extraction aims at locating and retrieving from the information sources the possible answers to the posed question. In Chapter 2 we reported that, to solve factoid questions, JUST.ASK relies on strategies that focus on the identification of answers through the use of regular expressions, gazetteers, statistical and WordNet-based recognizers. Here we introduce an approach to candidate answer extraction based on (lexico-syntactic) patterns, automatically learned from pairs of questions and their respective answers. This approach was inspired by our previous work on Question Generation (QG) that successfully used the learned patterns to generate questions (and their answers) from a target document, published in international conferences [Mendes and Coheur, 2011; Curto et al., 2011a,b] and the special issue of an international journal [Curto et al., 2012].

The present chapter is dedicated to the task of candidate answer extraction. It starts with a review of related work, followed by a description of our approach to pattern learning via a minimally supervised technique that takes pairs of questions and their answers as seeds. Afterwards, we detail how candidate answers are extracted, by explaining the unification process between the patterns and the sentences relevant to a question. Both the pattern learning and candidate answer extraction approaches are evaluated, their influence in JUST.ASK is measured and a discussion of the results is shown. A brief summary ends the chapter.

3.1 Related Work

For factoid-like questions, candidate answer extraction can be made as simple as extracting (groups of) consecutive tokens from text, discarding those that clearly do not agree with the posed question. This technique was implemented in the Aranea system [Lin, 2007], a

re-implementation of the AskMSR system by Brill et al. [2001]. Aranea extracts unigrams, bigrams, trigrams and tetragrams as candidate answers, and exploits heuristic knowledge to filter out those 'that are obviously wrong'. For instance, to questions that start with the phrase *How far* or *How tall* only n-grams with a numerical component are allowed; to questions that start with *Who* or *Where* a capitalized n-gram is required.

However, systems tend to prefer the use more linguistically informed approaches, for example based on NER. The underlying idea is the following: all named entities recognized in the information sources that agree with the answer expected type are candidate answers to the question. For instance, if a question expects as answer the name of an individual, all named entities of type PERSON recognized in the information sources are considered candidate answers. For this approach to work, it is required that the QA system discovers the type of information that is being sought after, typically through question classification, and the taxonomy of question categories has to be mapped into the taxonomy of named entity types. This is the approach currently followed by JUST.ASK and other systems [Mollá and Gardiner, 2004; Sarmento and Oliveira, 2007].

Another fairly used technique to candidate answer extraction relies on patterns. For example, the question *Who painted Guernica?* can be solved by matching the pattern "X was painted in 1937 by Y" with the sentence *Guernica was painted in 1937 by Picasso*, where X is an entity from the question (in this case, Guernica) and Y is the entity that answers it. The pattern-based approach to candidate answer extraction has received attention in QA specially after the TREC-10 competition, which consecrated as winner the system of Soubbotin [2001], entirely built around a list of hand-built surface patterns. Many works followed that rely on manually created patterns [Mollá and Gardiner, 2004; Hickl et al., 2006]. Nevertheless, despite the high-precision results achieved by the hand-built patterns, their coverage is often reported as very low; in addition, the creation of patterns is considered a time consuming, tedious and error-prone task, which often requires a human expert.

The focus has therefore turned to automatic pattern learning, which often takes advantage of information sources of large dimensions. Ravichandran and Hovy [2002] used a minimally

supervised approach to learn patterns for the purpose of candidate answer extraction and experimented a two-step method to acquire surface patterns of a certain class, given a set of seed examples composed of two entities: a question term and the answer. The first step acquires lexical patterns (from the Web) that relate a question term to the answer. For instance, to BIRTHYEAR questions the seed "Mozart 1756" is used to learn patterns like "<NAME> was born on <ANSWER>,". The second step validates the learned patterns by calculating their precision, a measure of the frequency of the pattern in the information sources. Other approaches that automatically learn patterns from large document collections are similar to, or based on, the work of Ravichandran and Hovy: relevant terms are extracted from the questions and used, with the answer, to query the information sources. Patterns are afterwards built from the retrieved relevant sentences. The system of Zhang and Lee [2002], called LAMP, uses questions, instead of two entities, and their answers (gathered from the TREC competitions). However, the seeds to the pattern learning algorithm are again composed of two entities: the answer and a question term, extracted by a set of regular expressions applied to the questions, which also determine their type. In respect to this, Schlaefer et al. [2006] argue that, on many questions, other important information is required to be present in the pattern. For instance, a pattern built from the question *How many calories are there in a Big Mac?* should contain the important terms "calories" and "Big Mac", besides the answer.[1] Their work (the Ephyra system) takes as input questions and answers, and learns patterns between the answer and two or more key-phrases. Again, these are extracted from the question by hand-made patterns. The approach of Du et al. [2005], however, resorts to a named entity tagger, a parser and a chunker to extract relevant terms from a question. Patterns are built with these terms and the answer to the question.

The extraction of candidate answers using patterns requires a literal match of the pattern with the sentence. Therefore, the number of extracted answers depends on: 1) the nature and content of the pattern; and, 2) the redundancy and diversity of the information sources. In particular, patterns that are too specific of an input seed are often ruled out by a validation step or may not match with the sentences retrieved from the information sources. Such

[1] Example taken from [Schlaefer et al., 2006].

problem is often circumvented with the introduction of higher levels of linguistic information in the pattern [Greenwood and Gaizauskas, 2003; Kosseim et al., 2003; Shima and Mitamura, 2010; Curto et al., 2012]. For example, by replacing certain surface words by their Part-of-Speech (PoS) or named entity tag: a pattern like "*X* was painted in 1937 by *Y*" is rephrased as "*X* was painted in <date> by *Y*", where the tag "<date>" abstracts the instance of date. Jijkoun et al. [2004] studied the influence of the presence of surface and syntactic information (by means of dependencies) on patterns and its overall impact on a QA system. Experiments were performed on the AQUAINT collection and authors report an increase in recall when using a higher layer of linguistic information, leading to the increment of the number of correct answers (assuming a robust selection of the final answer).

Shen et al. [2005] and Bouma et al. [2011] learn patterns based in syntactic dependency relations. A pattern is defined as the smallest dependency tree connecting two nodes (the answer and one question term). However, while the approach of Shen *et al.* uses tree kernel methods to compute the similarity between patterns and the parse trees of the sentences where answers might be stated, the approach of Bouma *et al.* is used offline to extract facts to populate a table (the Answer Extraction process is later accomplished by means of table look-up). Mollá [2006] uses the graph representation of dependency parse trees (of both questions and the sentences where the answers are stated) and builds answer extraction rules through graph manipulation. Kosseim and Yousefi [2008] add semantic information to the learned patterns. The learned patterns contain lexical, syntactic and semantic features (surface words, NP and named entity tags, respectively) and are augmented with a semantic constraint that indicates its semantics, based on the WordNet's hierarchy.

3.2 Pattern Learning via a Minimally Supervised Approach

A pattern relates a question to its answer through the constituents of the sentence that contains (parts of) the question and the answer. To learn patterns, we use a minimally supervised approach where seeds are Q/A pairs. For instance (*Who wrote The Divine Comedy?*, *Dante*)

is a Q/A pair.

The process of learning patterns comprises three different sequential steps:

1. the *Question Processing* step, where several actions are performed on the question, including its syntactic analysis and semantic classification;

2. the *Passage Retrieval* step, where multiple queries are built from the question and the answer, and used to retrieve passages from the information sources (the Web or local corpora); and,

3. the *Pattern Building* step, where the elements from the question, the answer and the retrieved passages are chosen to build the patterns.

In the following we detail each of these steps. Since retrieval is limited to the use of a search engine, when describing step 2, we will only focus on the process of building different queries.

3.2.1 Question processing

A (Q_i, A_i) pair is used to learn patterns. First, the question Q_i is classified into a semantic category that indicates the type of the expected answer (this corresponds to the task that was previously introduced as question classification). Then, the lexical and syntactic constituents of Q_i (L_i and S_i, respectively) are captured by exploring its parse tree. To maximize the number of learned patterns, different syntactic segments are captured, according to several heuristics. In particular, we apply tree cuts at different levels, iteratively breaking the phrasal nodes into its children nodes. As an example, consider the question *What is another slang expression for the Madness?*, whose parse tree is shown in Figure 3.1.

The analysis of the parse tree of this sentence leads to three different segmentations:

1. Fine-grained segmentation

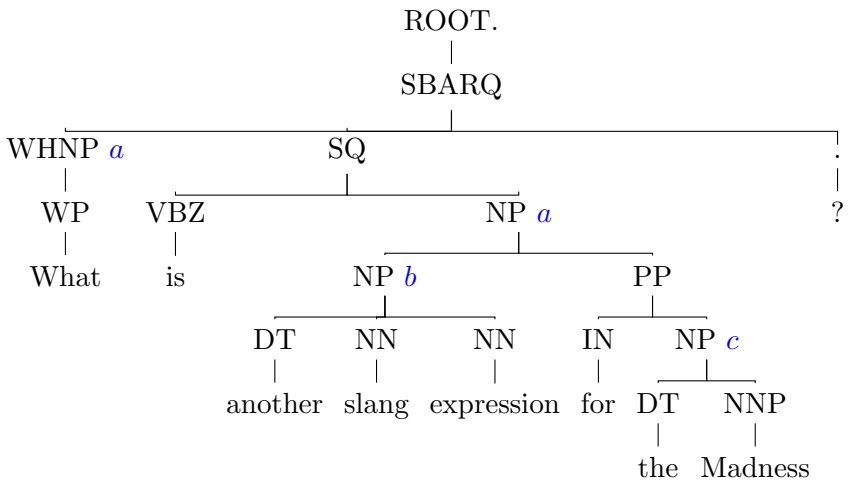


Figure 3.1: Parse tree of the question *What is another slang expression for the Madness?*

- [SYN-"WHNP VBZ NP$_b$ IN NP$_c$"]
- [LEX-"What, is, another slang expression, for, the Madness"]

2. Medium-grained segmentation

- [SYN-"WHNP VBZ NP$_b$ PP"]
- [LEX-"What, is, another slang expression, for the Madness"]

3. Coarse-grained segmentation

- [SYN-"WHNP VBZ NP$_a$"]
- [LEX-"What, is, another slang expression for the Madness"]

With the three distinct segmentations we are varying the amount of information from the question to be present in the learned patterns.

In should be clear that for some questions, the segmentation does not vary, regardless of the granularity level: *e.g. Who wrote The Divine Comedy?* will only result in [SYN-"WHNP VBZ NP"] and [LEX-"Who, wrote, The Divine Comedy"].

3.2.2 Passage retrieval

Multiple queries containing the seed answer A_i and subsets of L_i (except the one that corresponds to the WH-phrase in S_i) are created and submitted to the search engine.

For each question segmentation, three query types are created, containing:

Query Type 1 – the answer A_i and subsets of L_i. For example, `wrote "The Divine Comedy" Dante`;

Query Type 2 – the answer A_i and subsets of L_i, given that the main verb is conjugated in its different inflections and the auxiliary verbs (if they exist) are removed. For example, `Dante written "The Divine Comedy"`;

Query Type 3 – the answer A_i and subsets of L_i, except the verbs. For example, `Dante "The Divine Comedy"`;

All the passages retrieved for the queries are merged into a single data structure, with duplicates removed. Our motivation to perform multiple searches in the Passage Retrieval step is that it increases the number of relevant passages from where to build patterns. Furthermore, by adjusting the amount of information from the question to be contained in the queries, we allow patterns to be built from sentences that contain only parts of the question. An example is the pattern "NP? POS:'S NP", which does not contain reference to the main verb of the question. In both cases, the goal is to maximize the number of learned patterns.

All query types are used to retrieve relevant passages for a single question Q_i and, as we will see in Section 3.2.3.1, they will originate the three different types of patterns.

3.2.3 (Lexico-syntactic) Pattern building

Consider again the Q/A pair (*Who wrote The Divine Comedy?*, *Dante*) with the previous lexical and syntactic constituents, the query `Dante written "The Divine Comedy"` and the

sentence *In the 14th century Dante has written The Divine Comedy, considered the preeminent work of Italian literature.* retrieved as a search result, which has its parse tree in Figure 3.2.

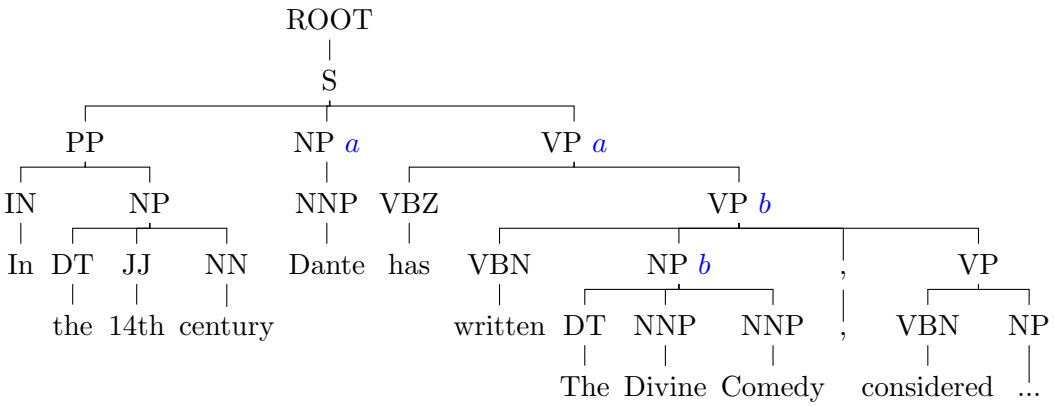


Figure 3.2: Parse tree of the sentence *In the 14th century Dante has written The Divine Comedy, considered the preeminent work of Italian literature.*

The following procedure builds the lexico-syntactic patterns:

1. Using regular expressions, identify the sequence(s) in the sentence that contain(s) all query elements, allowing one wildcard in between. In the example sentence, the regular expression "Dante .* written The Divine Comedy" identifies the sequence *Dante has written The Divine Comedy.*

2. In the subtree that conveys each identified sequence (Figure 3.3(a)):

   (a) Identify the leaf nodes that contain the answer A_i and the elements of L_i. In this case, the nodes "Dante", "written", "The", "Divine" and "Comedy" (Figure 3.3(b));

   (b) Identify the topmost nodes that contain uniquely each of the elements of L_i and the answer A_i. In this example, the nodes NP_a VBN NP_b (Figure 3.3(c)).

   (c) Identify the remaining leaf nodes. In this example, the node "has";

   (d) Collect the tags of the nodes identified in 2b. In this case, the tags NP VBN NP. Notice that these tags are the syntactic constituents of the pattern, representing an abstraction of the lexical components of the question and the answer;

(e) Collect the tags of the leaf nodes identified in 2c, and the tags of the nodes that contain each of them. In this example, the tag VBZ. Notice that these tags represent the context of the pattern, in particular the lexical tokens that are not in the seed question, nor in the answer;

(f) A pattern is created with n constituents, where n is the sum of the number of nodes identified in step 2b and 2c. A pattern is basically the sequence of the collected nodes, when the subtree is traversed in a depth-first fashion (see Figure 3.3(d)). In this example, the pattern "NP? VBZ:has VBN NP" is created.

(g) The correspondence between the elements of S_i and the syntactic constituents of the pattern is established by means of indices. In this example, the subscripted indices in the pattern "NP? VBZ:has VBN_1 NP_2" and in the syntactic segmentation of Q_i, [SYN-"WHNP VBZ$_1$ NP$_2$"] allow to make the relation.[2]

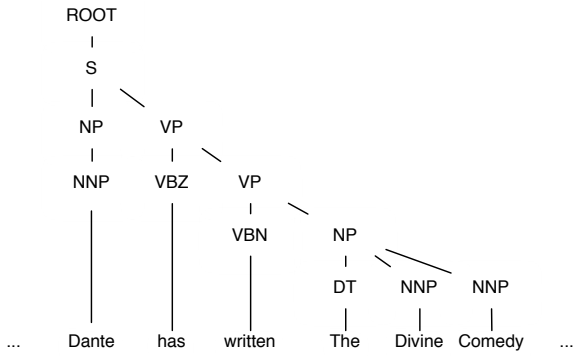
Thus, in this work we define a pattern as a sequence of lexical and syntactic constituents. All the learned patterns and the Q/A seed pairs are stored in a knowledge base, indexed by the semantic category and syntactic segmentation of the seed question.

3.2.3.1 Types of patterns

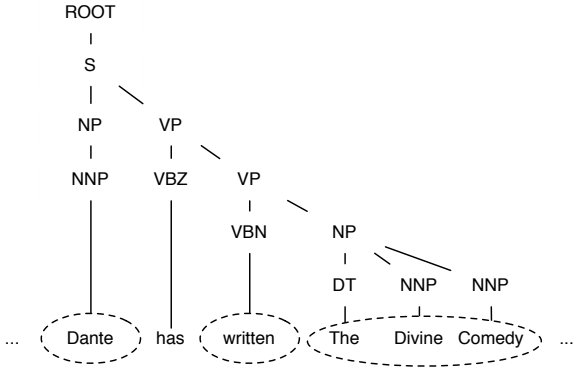
Given that we vary the content of the query used to discover the documents from which the patterns are learned, our approach is able to learn three types of patterns (such variability in the type of learned patterns was also explored in the context of question-generation in the THE-MENTOR system [Curto et al., 2012]).

1. STRONG patterns contain all the phrases (and their content) of the seed question. The query submitted to the search engine is built with the content of all phrases and the answer. For instance, "NP? VBD DT:the NN:poem NP" is a STRONG pattern learned from the sentence *Dante wrote the poem The Divine Comedy*;

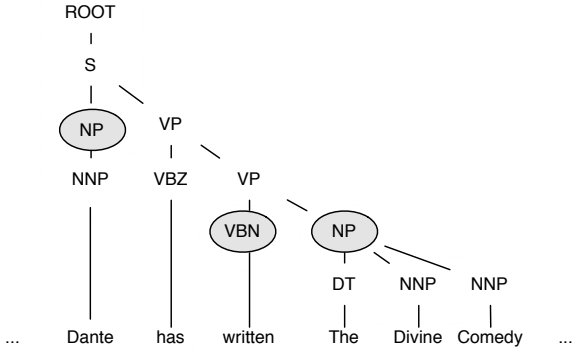
[2]The constituent that conveys the answer is marked with a question mark.



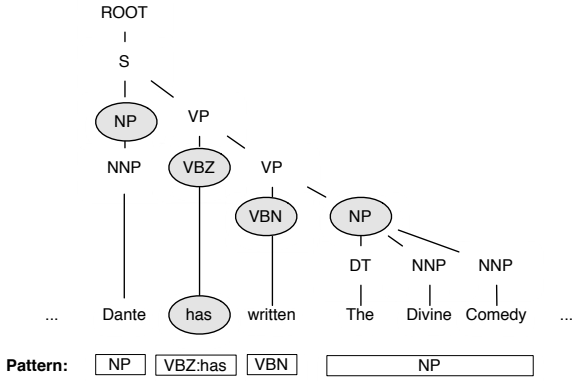
(a) The subtree that contains all the query elements, plus an extra token matched with a wildcard.



(b) The leaves of the subtree that contain the elements of L_i and A_i.



(c) Syntactic tags of the topmost nodes that contain the elements of L_i and the answer A_i



(d) The subtree nodes that compose a pattern.

Figure 3.3: Building a (lexico-syntatic) pattern from (the parse-tree of) a sentence that contains entities from the question and the question's answer.

2. INFLECTED patterns are learned after the inflection of the main verb of the question in its various tenses. The auxiliary verb is not required to be in the pattern. The query submitted to the search engine is built with an inflected form of the main verb of the question, the content of the remaining phrases and the answer. For instance, "NP? VBZ:has VBN NP" is an INFLECTED pattern learned from the sentence *Dante has written The Divine Comedy*;

3. WEAK patterns do not contain any verb phrase. The query submitted to the search engine is built with content of all phrases of the question, except the verb, and the

answer. For instance, "NP? POS:'s NP" is a WEAK pattern learned from the sentence *Dante's The Divine Comedy.*

The decision of learning different types of patterns resulted from the fact that, if patterns are too specific (STRONG patterns), they will not frequently match and not many answers will be found; if patterns are too generic (WEAK patterns), they will often match, but the appropriateness of the discovered answers is not certain.

When it comes to the WEAK patterns, it should be noticed that the relation between the entities is implicitly stated, since these patterns lack on the verbal constituent. Instead, in the STRONG and INFLECTED patterns, the relation is explicit, that is, we can capture the relation from the pattern since we know which of its constituents is associated with the verbal component of the seed question

3.3 Pattern-Based Answer Extraction

When a new question is posed to the system, a similar procedure to that of Pattern Learning is followed. That is, the question is first syntactically analyzed and semantically classified by the Question Processing component (as in Section 3.2.1), which is also responsible for extracting the question's lexical and syntactic segmentations. Afterwards, a set of relevant passages are retrieved by the Passage Retrieval component (as in Section 3.2.2), however here the formulated query does not contain the answer to the question. The next step is to extract candidate answers from those passages using the patterns, which we detail in the following.

3.3.1 Pattern/sentence unification

The classification of the new question, as well as its syntactic segmentations, allows to retrieve from the knowledge base all the applicable lexico-syntactic patterns. For example, consider the question *Who painted the Birth of Venus?* and its syntactic segmentation [SYN-"WHNP VBZ NP"]. The pattern "NP? VBZ:has VBN_1 NP_2" is retrieved from the knowledge base as being

applicable. The next step is to unify the pattern with the relevant sentences (those that are probable to contain the answer).

Given that we are dealing with patterns that convey lexical and syntactic informations, the unification of the patterns with a sentence is done both at the lexical and syntactic levels. For that purpose, we implemented a (recursive) algorithm that explores the parsed tree of a sentence in a top-down, left-to-right, depth-first search, unifying the constituents of the parsed text with the linguistic information in the patterns.

The algorithm for pattern/sentence unification goes as follows:

1. Tests if the sequence of syntactic constituents of the pattern exists in the sentence. This means that, in a depth-first search of the tree, the sequence of syntactic tags in the pattern has to occur (consecutively or not, given that we allow gaps between the nodes). If so, collects that information chunk and continues; if not a relaxed strategy is applied;

2. If the pattern does not contain lexical information, continues to (4);

3. If the pattern contains lexical information, tests if it matches the tokens in the leaves of the parse tree of the text segment, in the same relative position. If so, continues;

4. Return the text segment, since there is a lexico-syntactic overlap between the pattern and the text segment.

Again, for illustration, consider the pattern "NP? VBZ:has VBN_1 NP_2" and the sentence *With the sponsorship of the Medici family Botticelli has painted the Birth of Venus*, whose parse tree is depicted in Figure 3.4.

There are 12 text segments that match the syntactic information present in the learned pattern. They result from the all the possible combinations of the different NP's before and after the VBZ VBN sequence:

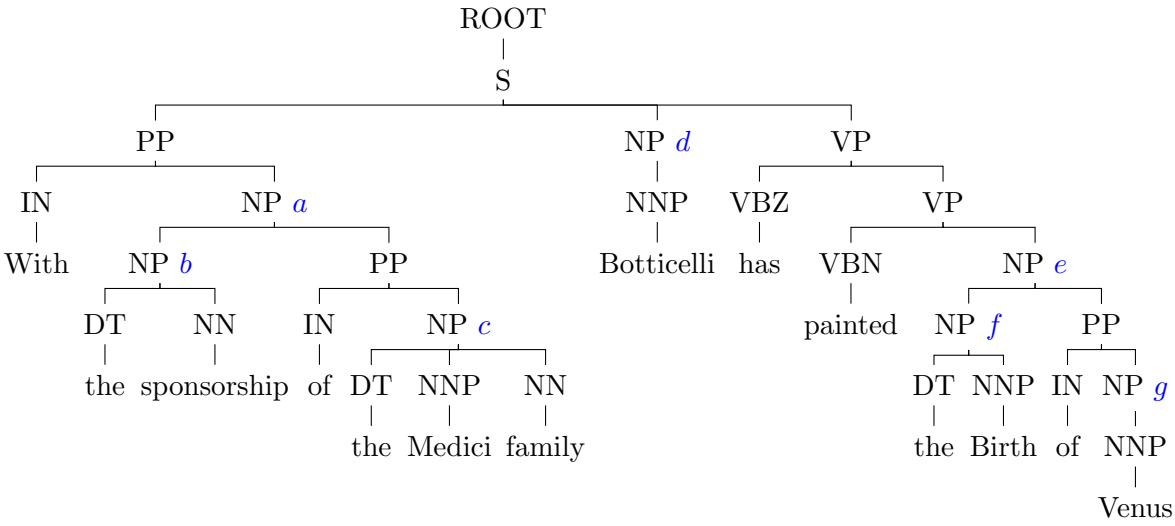


Figure 3.4: Parse tree of the sentence *With the sponsorship of the Medici family Botticelli has painted the Birth of Venus.*

- *the sponsorship of the Medici family has painted the Birth of Venus* (conveyed by nodes NP_a VBZ VBN NP_e);
- *the sponsorship of the Medici family has painted the Birth* (conveyed by nodes NP_a VBZ VBN NP_f);
- *the sponsorship of the Medici family has painted Venus* (conveyed by nodes NP_a VBZ VBN NP_g);
- *the sponsorship has painted the Birth of Venus* (conveyed by nodes NP_b VBZ VBN NP_e);
- *the sponsorship has painted the Birth* (conveyed by nodes NP_b VBZ VBN NP_f);
- ...

At this point, there is a lexico-syntactic overlap between the pattern and every one of the extracted text segments.

The goal of the unification is to recover from the sentence the information chunk that represents the answer to the posed question. Therefore, the next step is to assure that the

constituents of the extracted text fragment match the respective constituents of the posed question. For that purpose, we compare the lexical constituents of the new question with the lexical constituents of each extracted text fragment, using the associations between the pattern and the source question. That is, from the pattern "NP? VBZ:has VBN_1 NP_2" we know that the last NP refers to the 2^{nd} position of the question's segmentation ([SYN-"WHNP VBZ NP"] and [LEX-"Who, painted, the Birth of Venus"]). Therefore, we rule out all the text segments that do not convey *The Birth of Venus* in the last NP.

Finally, the words in the text fragment associated with the pattern constituent relative to the answer (marked with a question mark) is returned as the answer to the posed question.

Note that, if multiple unifications of a pattern in one sentence occur, we extract multiple candidate answers from that sentence. In our example, *Botticelli*, *the sponsorship of the Medici family*, *the sponsorship* and *the Medici family* are returned as candidate answers.

3.3.2 Relaxing the unification

The inherent variability of natural language often precludes a strict pattern/sentence unification. A challenge in pattern-based approaches to Answer Extraction is, indeed, to understand the linguistic level at which the unification should be performed [Shima and Mitamura, 2010]. To cope with this issue, we introduce a set of (lexicon, syntactic and semantic) relaxations, that aim at easing the unification process, allowing the system to extract more instances of answers.

Therefore, if there are patterns in the knowledge base that apply to the question, but the unification does not succeed or no correct answer is captured, the system relies in the following relaxation strategies:

1. A lexical distance (Levenshtein and Jaccard) is permitted between the sentence and the lexical content of the pattern. This allows, for instance, to cope with orthographic errors and, for instance, to extract the answer to *Who wrote The Divine Comedy?* in a sentence such as *Dante has written The Divine* ***Commedy***.

2. A syntactic match is consented between (syntactic) classes that relate with each other as they belong to the same (super) class (*e.g.*, both "NP" (proper noun) and "NNS" (plural noun) are nouns). This allows the pattern "NP [has] VBN NP" to match a sentence analyzed as "NN [has] VBN NP".

3. A semantic match is allowed between two words, if there is a synonym relation between them stated in WordNet. This allows a pattern "NP VBD [established in] NP" to match a sentence *Habitat for Humanity was founded in 1976.*

The system's performance cascades over five different performance levels, from the strictest to the loosest, where the unification is continuously relaxed (for instance by allowing a higher lexical distance between the pattern and the sentence).

3.4 Evaluation

In this section, we present the experiments conducted to evaluate the pattern-based approach to candidate answer extraction and we show the achieved results.

We first report the number of patterns learned, afterwards we use the pattern-based approach and show how it contributes to the candidate answer extraction in JUST.ASK; finally we discuss the influence of the pattern-based strategy on the system's overall results, when it is used in conjunction with the NE-based strategies implemented in JUST.ASK.

The TREC-QA_2002-2007 corpus was used in all three experiments and, although there is typically more than one correct answer for the questions in the corpus, a Q/A pair used to learn patterns is always composed of the question and its first (correct) answer, unless otherwise stated.

3.4.1 Pattern learning

A total of 195,787 patterns – 4,148 `strong`, 28,150 `inflected` and 163,489 `weak` – were learned for 1,263 questions, 88% of the original set of 1,440 questions. That is, no patterns

were learned for 177 questions, which is explained by the following reasons (the typical scenario is all of the points contributing to the failure of the pattern-learning approach, not just one):

- the search engine did not retrieve any result to the queries formulated for a given question, which happen mostly in queries composed of several query terms (`"12 500 miles" and "by" and "established" and "the boeing 777" and "the non-stop distance record"`);

- not all of the (exact) terms of the query could be found in the search result and, therefore, the patterns could not be built. Recall that our approach requires that all of the query terms should be in the retrieved search result; however, in many queries that, for example, contain stop words (`"in annapolis" and "naval academy"`) or question words (`"author jasper fforde's" and "who killed humpty dumpty?"`), the search engine might simply omits those words in the results.

- there was not context between the query terms in order to a pattern to be built. Again, remember that the learned patterns always contain context words to connect the question parts to the answer and, therefore, a pattern can not be built from a sentence *the naval academy in annapolis* retrieved for the query `"in annapolis" and "naval academy"`, for instance.

In this experiment, 50 passages were asked to the search engine from where to learn patterns (per query). Table 3.1 shows the number of learned patterns per coarse category and also indicates the number of questions that led to patterns in each category.

The number of patterns learned seem to depend both on the number of existing questions and on the category of the questions. That is,

- the more questions exist, the more patterns are learned. For example, the number of patterns learned for category ENTITY is much higher that the number of patterns learned for category ABBREVIATION or DESCRIPTION;

		Learned Patterns			
Category	# Q/A pairs	`strong`	`inflected`	`weak`	Total
ABBREVIATION	18 (85.7%)	24	394	2.352	2,770
DESCRIPTION	81 (81.0%)	408	2,498	8,481	11,387
ENTITY	191 (93.6%)	493	3,797	30,481	34,771
HUMAN	269 (90.3%)	523	4,700	37,082	42,305
LOCATION	264 (90.1%)	1,813	8,898	46,496	57,207
NUMERIC	440 (84.0%)	887	7,863	38,597	47,347
Total		4,148	28,150	163,489	195,787

Table 3.1: Distribution of the number of learned patterns per (coarse) category.

- some question categories are more likely to represent good seeds. For example, although it is not the most populated category in terms of questions, the category LOCATION lead to the higher number of learned patterns. We believe that this is mostly related with the lexical variation allowed in the answers of these categories: for example, there are fewer ways to write the name of a location that to of a date.

Moreover, while nearly half of the `strong` patterns were learned for category LOCATION, the number of learned patterns per category seem more well distributed in the other types of patterns.

Figure 3.5 shows the pattern distribution per (fine) question category. For a better visualisation, we set the unit scale to Log (logarithmic). The highest number of patterns were learned for category HUMAN:INDIVIDUAL, while the questions of category HUMAN:DESCRIPTION generated the fewer patterns. No patterns were learned for 2 categories (ENTITY:TECHNIQUE and NUMERIC:WEIGHT). The question *What kind of plant is kudzu?* from category ENTITY:PLANT led to the higher number of `strong` patterns, category LOCATION:MOUNTAIN led to the higher ratio of learned `inflected` patterns/question and category ENTITY:CURRENCY to the higher ratio of `weak` patterns/category. No `strong` patterns were learned for 11 categories.

We calculated the number of learned patterns if the seed Q/A pairs were built using the questions and all their respective pairs in the TREC-QA_2002-2007 corpus (*i.e.*, considering all questions and all answers in the corpus). The results are the following: a total of 336,518

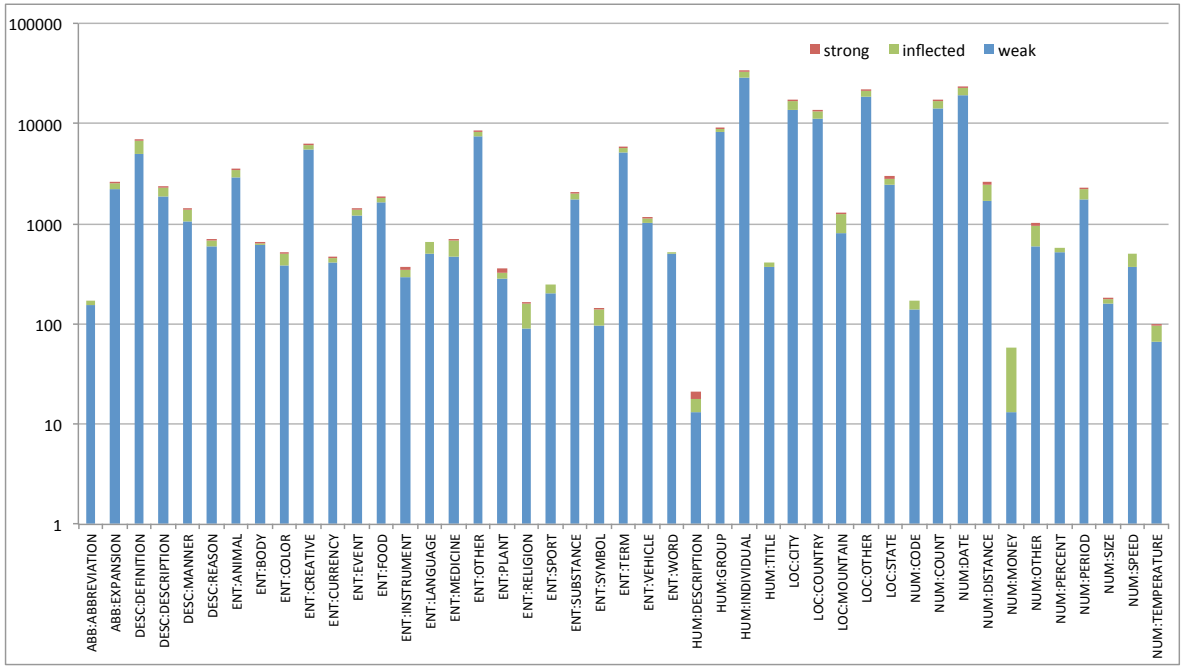


Figure 3.5: Number of patterns learned per question category.

patterns – 7,449 `strong`, 49,727 `inflected` and 279,342 `weak` – were learned for 1318 questions. The number of questions in the set of 1,440 that led to a pattern increased to 94% (1,359 questions), pointing out the importance of choosing a good Q/A pair as the seed of the pattern learning algorithm. From the 96 new questions that led to patterns, 17 belonged to the coarse category HUMAN and 52 to the coarse category NUMERIC.

3.4.2 Pattern/sentence unification

3.4.2.1 Experiments

To understand the impact of the pattern/sentence unification in the answering process of JUST.ASK, we conducted two different experiments. The difference between the two lies on the usage of different training and testing sets, in particular:

Experiment 1: N-fold cross validation – The corpus is shuffled and split into N parts: patterns are learned from the Q/A pairs of the $N-1$ parts (the training set) and

applied to answer the questions of the remaining part. In our experiments we used $N = 6$, therefore in every portion of the corpus there are 240 questions.

Experiment 2: Yearly evolution – Since the TREC-QA_2002-2007 corpus is composed of the corpora used in sequential editions of the TREC competition, we answer the questions of the edition of year i using as training set the Q/A pairs from the years till year i. In this way, we aim at knowing how many questions we would be able to answer in one year if we used as training the questions and answers from the past.

In both experiments, we vary the number of passages asked to the search engine from where to learn patterns (which we denote MAX_{pl}) and the number of passages asked to the search engine from where to extract answers (which we denote MAX_{ae}). We report the number of correct answers extracted by JUST.ASK and we make two important assumptions:

- the passage retrieval component only retrieves passages that contain at least one correct answer for a given question; and,
- the answer selection step is flawless, that is, if JUST.ASK is able to extract a correct answer, it will return it.

Regarding the JUST.ASK pipeline, we disabled the normalisation, clustering and filtering phases in the final answer selection step (see Subsection 2.2.3.2). Only the candidate answer aggregation step is performed and the list of final answers is ordered by decreasing frequency of the answers. If no answer is found in the list of candidate answers the system's behaviour depends on whether the relaxation strategies are enabled or not: if so, the performance level of the system worsens and system loosens its unification constraints; if not, the system leaves the question unanswered.

3.4.2.2 Results

3.4.2.2.1 Experiment 1 – N-fold cross validation As previously stated, in this experiment we split the TREC-QA_2002-2007 corpus in N even parts and used $N-1$ as training

set and the remaining as test set (we set $N = 6$).

Regarding the system's performance when varying MAX_{pl} and MAX_{ae}, Table 3.2 shows the number of correct questions according to different values of MAX_{pl} and MAX_{ae}, when the pattern-based approach is applied with ($WRelax$) and without ($WoRelax$) relaxation. It also presents the number of patterns learned for each value of MAX_{pl}.

MAX_{pl}	**5**			**10**			**25**			**50**		
MAX_{ae}	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**	**25**	**50**	**100**
#Patterns	22,605 ± 212			41,951 ± 370			91,661 ± 932			163,492 ± 1,777		
$WoRelax$	24	36	47	27	38	51	29	43	55	31	45	57
	10%	15%	20%	11%	16%	21%	12%	18%	23%	13%	19%	24%
$WRelax$	64	83	98	70	90	106	75	96	113	80	101	118
	26.8%	34.6%	40.8%	29.1%	37.3%	44.2%	31.2%	39.9%	46.9%	33.5%	41.9%	49.3%

Table 3.2: Number of correct questions and recall when varying the values of MAX_{pl} and MAX_{ae} in Experiment 1.

From the achieved results, we can see that:

- the system's recall positively correlates with the number of passages from where to learn patterns and with the number of passages from where to extract answers. However, the system's performance is much more dependent of the number of passages from where to extract answers, than of the passages from where to learn patterns. These results were also verified in previous experiments [Mendes et al., 2013a] where patterns were learned and answers were extracted from a document collection, locally indexed; and,

- the system's recall is higher if the relaxation strategies are used, as expected. In all the cases, the use of relaxation doubles the system's recall results.

It is natural that the use of more information, both in terms of patterns and passages from where to extract answers, lead to an higher number of extracted candidates answers. However, and although the system is able to extract more correct answers (as it is shown in the tables), it can also signify an increase of the number of extracted incorrect answers, which the system has to deal with in downstream components of the pipeline (the answer selection component). Therefore, it is important to verify if there is a balance between correct and

incorrect extracted answers or, at least, if the number of extracted incorrect answers does not increase in a higher proportion when compared with the number of extracted correct answers, leading to a drop of the system's precision and MRR. The charts in Figures 3.6 and 3.7 allow us to understand the variation of the system's precision and MRR depending of the values of MAX_{pl} and MAX_{ae} and of the use (or not) of the relaxation strategies. The MRR is calculated uniquely for the answered questions.

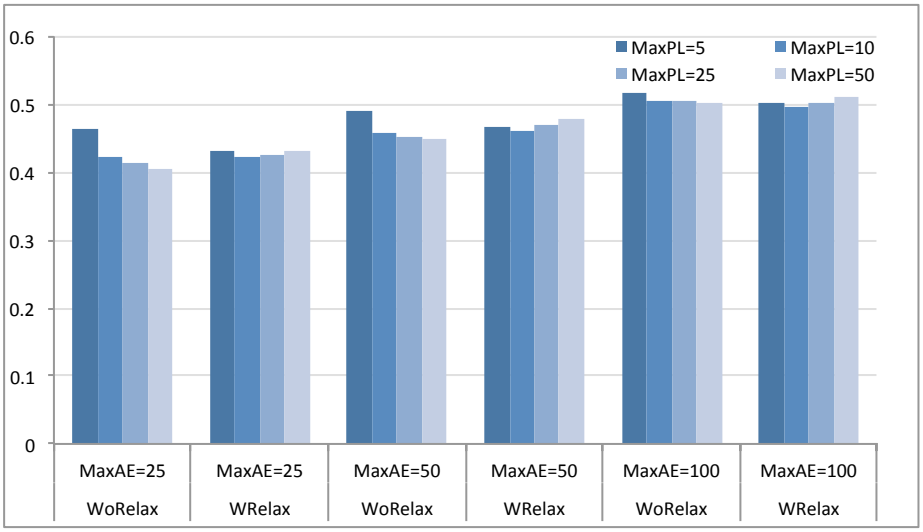


Figure 3.6: Precision when varying the values of MAX_{pl} and MAX_{ae}, with and without relaxation, in Experiment 1.

Regarding JUST.ASK's precision, the use of more patterns slightly prejudices the system's precision for fixed values of MAX_{ae}, when the relaxation strategies are not applied. However, when the relaxation strategies are used, the precision increases for values of MAX_{pl} higher than 10. Moreover, the use of more passages from where to extract answers leads to an increase of the precision: the best precision results are attained when MAX_{ae} has its higher values.

The achieved precision results show that, although more answers (correct and incorrect) are being extracted by the system, the ratio between correct and incorrect questions does not vary with the use of more information in the pattern-based approach to candidate answer extraction.

Results are similar regarding the system's MRR for the answered questions. We can see

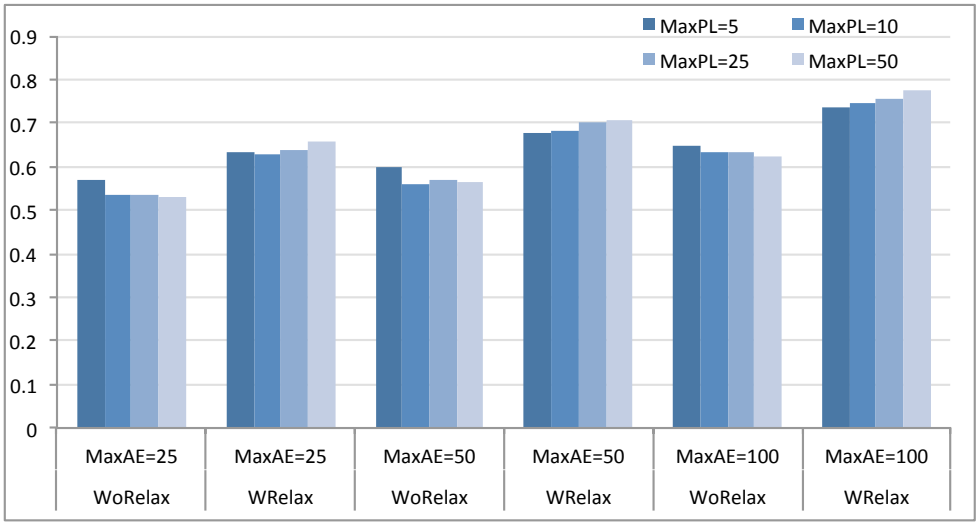


Figure 3.7: MRR when varying the values of MAX_{pl} and MAX_{ae}, with and without relaxation, in Experiment 1.

that the MRR decreases with the increase of patterns for fixed values of MAX_{ae}, when no relaxation is used. However, for equal values of MAX_{ae} and MAX_{pl}, the use of relaxation has a clear positive impact in the system's MRR. Again, as it occurs with the precision, the use of more passages from where to extract answers leads to an increase of the MRR, the best results being attained for the higher value of MAX_{ae} and MAX_{pl}.

The achieved MRR results suggest that the use of more information allows the system to extract and to position the correct answers in higher ranks of the list of returned answers.

3.4.2.2.2 Experiment 2 – Yearly evolution In this experiment, we evaluate how the system behaves when learning to answer using the available information from the past, by testing three different configurations. In the following, we describe each of the configurations and present the achieved results.

For a better analysis of the results, in Table 3.3 we present some statistics of the TREC-QA_2002-2007 corpus, as well as some intermediate results achieved by JUST.ASK. In particular, for each year, we show the number of test questions, the number of different question categories and the number of questions with at least one positive passage retrieved by the search engine. Regarding the question categories, and to evince the differences between the

years, we show the number of categories shared by both years (in between parentheses).

TREC year	2002	2003	2004	2005	2006	2007
# Questions	447	378	66	141	209	199
# Categories	36	37 (23)	20 (16)	27 (11)	29 (9)	33 (15)
(cumulative)	36	43	43	44	46	47
% PosPassages	89.04%	83.60%	84.85%	80.85%	77.51%	84.93%

Table 3.3: Some statistics from the TREC-QA_2002-2007 corpus.

Configuration a) To answer questions of year i, only the Q/A pairs from year $i-1$ are used as training set. Q/A pairs are built with each question and the respective first correct answer.

Year	**2003**		**2004**		**2005**		**2006**		**2007**	
MAX_{ae}	**100**		**100**		**100**		**100**		**100**	
MAX_{pl}	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**
#Patterns	41,844	75,038	34,764	61,984	3,601	6,429	7,667	13,497	13,452	24,028
$WoRelax$	44	49	12	12	19	20	25	26	21	22
	11.6%	13.0%	18.2%	18.2%	13.5%	14.2%	12.0%	12.4%	10.6%	11.1%
$WRelax$	122	127	25	27	35	35	46	43	44	47
	32.3%	33.6%	37.9%	40.9%	24.8%	24.8%	20.6%	22.0%	22.1%	23.6

Table 3.4: Number of correct questions and achieved recall in Experiment 2, configuration a).

Table 3.4 shows the number of correct questions and the recall achieved in Experiment 2, configuration a). For each year, it also shows the number of learned patterns that are available for candidate answer extraction.

As it can be seen, if we were to compete in a TREC evaluation using uniquely our pattern-based approach to candidate answer extraction with patterns learned from the previous year's questions, we would correctly answer, at most, 18.2% questions (without relaxation) and 40.9% (with relaxation). This would have happened in the year 2004, with training set from year 2003. Year 2004 benefited from the large amount of patterns learned from the previous year, as well as from a semantic similarity of the test questions in both years: 80% of the question categories in year 2004 existed in year 2003, allowing the approach to learn patterns applicable for the new questions (see Table 3.3).[3]

[3]Results would be lower, as the number of test questions was higher in the competition. These calculations are based on the questions we have from that competition.

From the table, one can observe two main phenomena:

- a high number of learned patterns does not necessarily signify a good recall result, if no relaxation is used. This can be seen in the year of 2003, which had the highest number of available patterns, however it scored 2^{nd} worst in terms of recall results.
- a low number of learned patterns does not necessarily signify poor recall result. As it can be seen, year 2005 has the lowest number of learned patterns, however it was the year with the 2^{nd} best results, when no relaxation was used.

Configuration b) To answer questions of year i, the questions from years $i - j$ ($1 <= j < i$) are used as training set. Q/A pairs are built using the questions and the first correct answer. Q/A pairs are built with each question and the respective first correct answer.

Year	**2003**		**2004**		**2005**		**2006**		**2007**	
MAX_{ae}	**100**		**100**		**100**		**100**		**100**	
MAX_{pl}	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**
#Patterns	41,844	75,038	76,078	136,120	79,582	142,390	87,072	155,604	100,341	179,339
$WoRelax$	44	49	18	18	34	35	37	41	29	31
	11.6%	13.0%	27.3%	27.3%	24.1%	24.8%	17.7%	19.6%	14.6%	15.6%
$WRelax$	122	127	28	30	63	65	67	73	77	80
	32.3%	33.6%	42.4%	45.5%	44.7%	46.1%	32.1%	34.9%	38.7%	40.2%

Table 3.5: Number of correct questions and recall achieved in Experiment 2, configuration b).

Table 3.5 shows the number of correct questions and the recall achieved in Experiment 2, configuration a). For each year, it also shows the number of learned patterns that are available for candidate answer extraction.

If we were to compete in a TREC evaluation using uniquely our pattern-based approach to candidate answer extraction with patterns learned from all previous years' questions, we would correctly answer, at most, 27.3% questions (without relaxation) and 46.1% (with relaxation). This would have happened in the years 2004 and 2005, respectively.

Comparing with the results achieved in Experiment 2, configuration a), we can see that all recall results improve. Learning from all the questions seen till year n increases the recall

results in around 5% in average (when no relaxation is applied) and in more than 10% in average (when relaxation is applied). However, we can also report an increase of the learned patterns. This is due to a higher number of applicable patterns that were learned previously in the patterns.

Configuration c) To answer questions of year i, the questions from years $i - j$ ($1 <= j < i$) are used as training set. Q/A pairs are built with each question and using the questions and all the respective correct answers.

Year	**2003**		**2004**		**2005**		**2006**		**2007**	
MAX_{ae}	**100**		**100**		**100**		**100**		**100**	
MAX_{pl}	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**
#Patterns	76,102	135,555	131,812	234,672	138,784	246,860	152,041	270,177	171,217	304,180
$WoRelax$	50	55	18	19	38	38	39	42	30	33
	13.2%	14.6%	27.3%	28.8%	27.0%	27.0%	18.7%	20.1%	15.1%	16.6%
$WRelax$	131	137	31	33	67	68	71	75	82	85
	34.7%	36.2%	47.0%	50.0%	47.5%	48.2%	34.0%	35.9%	41.2%	42.7%

Table 3.6: Number of correct questions and recall achieved in Experiment 2, configuration c).

Table 3.6 shows the number of correct questions and the recall achieved in Experiment 2, configuration a). For each year, it also shows the number of learned patterns that are available for candidate answer extraction.

If we were to compete in a TREC evaluation using uniquely our pattern-based approach to candidate answer extraction with patterns learned from all previous years' questions, we would correctly answer, at most, 28.8% questions (without relaxation) and 50.00% (with relaxation). This would have happened, again, in the year of 2004.

As it occurred in the previous experiment, all recall results improved due to existence of more patterns. However, and as we have concluded from the results achieved in Experiment 1, recall does not seem to gain much with the increase in the number of passages from where to learn patterns (MAX_{ae}): although the number of patterns increases dramatically, carrying consequences in time and computational requirements, the number of extracted correct answers increases marginally (in fact, for some years, the results remain unchanged).

3.4.3 Influence in Just.Ask

This section is dedicated to the analysis of the influence of the pattern-based approach to candidate answer extraction on the overall performance on JUST.ASK, given that all the NE-based strategies are enabled (see Section 2.2.3.1 for a description of the strategies).

In this experiment, we used the configuration b) of Experiment 2: that is, $MAX_{ae} = 100$, the relaxation strategies are enabled and Q/A pairs are built using the questions and the first correct answer. Although the best results of the pattern-based approach were achieved in configuration c) of Experiment, where the Q/A pairs are built using the questions and all their correct answers, we decided to use what is common in the literature [Ravichandran and Hovy, 2002; Zhang and Lee, 2002; Schlaefer et al., 2006], where a seed is composed with elements from the question and one correct answer.

Year	**2003**		**2004**		**2005**		**2006**		**2007**	
MAX_{ae}	**100**		**100**		**100**		**100**		**100**	
MAX_{pl}	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**	**25**	**50**
NE-*based*	193		36		80		115		100	
(baseline)	51.1%		54.5%		56.7%		55.0%		50.3%	
NE + *Pattern*	200	201	37	37	82	82	117	117	103	105
WoRelax	52.9%	53.2%	56.1%	56.1%	58.2%	58.2%	56.0%	56.0%	51.8%	52.8%
NE + *Pattern*	225	226	37	39	92	92	125	127	116	117
WRelax	64.7%	64.9%	56.1%	59.1%	65.2%	65.2%	59.8%	60.8%	58.3%	58.8%

Table 3.7: Number of correct questions and associated recall when using uniquely the NE-based strategies and when using the NE-based strategies plus the Pattern-based strategy to candidate answer extraction.

As it can be seen from Table 3.7, the usage of the pattern-based approach to candidate answer extraction increases JUST.ASK's recall by from 1% to 2.5% (absolute values), when no relaxation is used. Although this is a very marginal increase, in the cases where relaxation is used, we see improvement of the recall augments in the order of 13.8% (absolute values).

3.5 Discussion

In this section we described our pattern-based approach to candidate answer extraction, presented several experiments and the achieved results.

The differences between our and other pattern-based approaches to candidate answer extraction can be summarized in the following:

Input seeds Our seeds are composed of questions and answers, and not two entities [Ravichandran and Hovy, 2002; Bouma et al., 2011]. Moreover, instead of extracting only the relevant terms from each question [Zhang and Lee, 2002; Schlaefer et al., 2006], we use the entire question (except the WH-phrase) and manipulate it in order to learn the maximum number of patterns;

Open-domain patterns Although applied to the extraction of short, fact based answers, our approach is not restricted to particular question or pattern types (like in Zhang and Lee [2002] and Schlaefer et al. [2006]);

Use of syntactic information Like in Shen et al. [2005] and Bouma et al. [2011], patterns that convey syntactic information are learned. However, we use syntactic chunks instead of dependencies. Moreover, our patterns are sequences of lexico-syntactic symbols, and not trees.

A common denominator to all the presented experiments has to do with the system's results depending whether the relaxation strategies are employed or not. We could see improvements in the system's recall of more that 20% when using relaxation compared with when no relaxation was allowed. The variety of the natural language prevents many patterns to unify with the sentences where the possible answers might lie, as the pattern-based approach to candidate answer extraction requires a literal match between the pattern and the sentence content. To overcome this limitation, we apply a set of relaxation strategies which proved to help to increase the system's recall.

Regarding the variation of MAX_{pl} and MAX_{ae}, we could see that the system depends more on the number of passages from where to extract answers, than on the number of passages from where to learn patterns: the increase in MAX_{pl} leads to a (much) higher number of learned patterns, which often do not lead to a significant increase of the system's recall. We consider that a balance between these parameters should be achieved that considers not only the number of correct questions and the system's MRR and precision, but also the computational power and the time required to learn patterns and unify them with the sentences.

To motivate for the importance of a question classifier in a QA system pipeline we reported that "*a misclassified question can hinder the ability of the system to reach the right answer*" (Section 2.1) and when discussing the strengths and witnesses of JUST.ASK, in particular the system's question classifier, we stated that "*no question incorrectly classified is correctly answered*" (Section 2.6). If one has an extractor that forces the extracted answer to match the semantics of the question, the previous considerations are certainly true. However, in our pattern-based this constraint does not apply: we use the semantics of the question as a mean to (semantically) group and index the learned patterns, but do not force an answer to belong to a certain category. This is the reason why the pattern-based approach was useful in the discovery of answers to misclassified questions. For instance, the question *What prison is Charles Manson at?* expects a location as answer. However, since it is classified as requiring the definition of something, the correct answer *Corcoran State Prison* is not extracted by the NER strategies. Patterns were successful in this situation and also to find answers to questions like *What is the "Playboy" logo?* and *How did Malcolm X die?*, for which JUST.ASK does not have a dedicated NE-based strategy.

The achieved results of the pattern-based approach strategy are still not comparable to the best results achieved in QA (in TREC 2007 the best system achieved a recall of 70%[4] [Dang et al., 2007]). One reason that contributed to the system's low recall results was the diversity of the posed questions. The total 1440 questions belonged to 47 different semantic categories

[4]Note that, however, this is not a fair comparison, as the test corpora and the information sources are different.

(from [Li and Roth, 2002]) and, for many categories, there were not enough questions to serve as seeds to pattern learning.

3.6 Summary

In this section we presented an approach to candidate answer extraction based on (three types of) patterns learned from pairs composed of a question and its answer. For that purpose we use a minimally supervised learning strategy to pattern learning, which we also described in this chapter. To extract candidate answers to questions, patterns are unified against the sentences retrieved as relevant by the Passage Retrieval module. Given that the variability of the natural language often precludes a strict pattern/sentence unification, we adopted a series of linguistic relaxation strategies, which were also described. We evaluated this approach in several experiments where, for instance, we answered the questions of a certain year of TREC using patterns learned from the questions (and answers) of past editions of the competition.

When compared to the results achieved by the NER-based strategies to Answer Extraction, the performance of the pattern-based approach is much inferior. However, the conducted evaluation allowed us to conclude that the pattern-based approach allows to extract correct answers to two different types of questions that JUST.ASK was not able to solve before: misclassified questions and questions for which there is no appropriate NER strategy.

4 Final Answer Selection based on Semantic Relations

The task of choosing the final answer(s) from a pool of candidate answers is known as final answer selection. In JUST.ASK this task is accomplished through a sequence of sub-tasks, where the frequency of each answer plays a determinant role. One of the downsides of this approach is that the candidate answers are seen as competitors and the semantic relations between them are not considered. In this chapter we show how these relations can contribute for a better final answer selection, using an approach that we published in an international conference [Mendes and Coheur, 2011].

The present chapter is dedicated to the task of final answer selection. It starts with a review of related work, followed by a description of our approach to answer selection based on semantic relations. Afterwards the evaluation is presented along with the achieved results, including the impact of the approach in JUST.ASK. Lastly, we make a brief discussion of the results and finish with a short summary.

4.1 Related Work

4.1.1 Techniques for final answer selection

Literature on QA shows examples of several different techniques for final answer selection. In the following we divide these techniques into two major groups:

Filtering/validation of candidate answers: A first group of techniques to answer selection aims at discarding (or filtering) the wrong candidate answers, while keeping (or validating) the correct ones.[1]

[1] In QA these two tasks – filtering and validating candidate answers – are often viewed as similar; however,

When it comes to filtering, many techniques are built on the assumption that candidate answers that agree with certain constraints (inferred from the analysis of the question) are possibly correct, while those that do not agree with the constraints are certainly wrong. These constraints lie at different linguistic levels:

Lexical The Aranea system [Lin, 2007] used a set of heuristic rules to filter out n-grams that are 'obviously wrong' answers. For instance, only n-grams with a numerical component are allowed to questions that start with the phrases *How far* or *How tall*; to questions that start with *Who* or *Where* a capitalised n-gram is required. All the other candidate answers are rejected. Other constraints imposed by Aranea at the lexical level are related to the information overlap between the posed question and candidate answer: if the latter is contained in the former, it is also discarded.

Syntactic The system of Cardie et al. [2000] only considered noun phrases as candidate answers, while all the other syntactic constituents are excluded from being final answers, which lead to a slight improvement on the achieved results.

Semantic Schlobach et al. [2007] presented an extensive work on semantic filtering of candidate answers, by using different knowledge-(and redundancy-)based methods. Filtering based on semantics usually presupposes a question classification step, where the question is annotated with information about the expected answer type (see Section 2.1 for some details on question classification).

Regarding answer validation, Harabagiu and Maiorano [1999] define it as an inference process that considers an answer to a question to be valid if an explanation for it can be found. Moldovan et al. [2003a] introduced COGEX, a logic prover that aims at understanding the relations between the question and each candidate answer through linguistic and world knowledge axioms, and performs logical inference to validate the candidate answers. The use of COGEX for answer validation led to an increase of

each concept carries a distinct semantic meaning that is important to mention. While filtering is associated with ruling out the wrong answers, validation is typically related with retaining the correct ones.

performance of 30% on the achieved results. In this perspective, answer validation is a highly knowledge-dependent task; however, in open-domain QA systems that search for answers in large-scale information sources, knowledge intensive strategies for answer validation are deemed impracticable, due to the need of intensive computational processing (besides the need of a very large-scale knowledge repository).

A solution to tackle the problem of answer validation in open-domain QA is again to take advantage of the statistical properties of large amounts of data. For instance, Magnini et al. [2002] see answer validation from a data-driven point of view and assume that a candidate answer to a question is valid if the question and the answer often co-occur. First, several keywords are extracted from the answer and question and, with those, a reformulation pattern is created, which is afterwards submitted to the Web. The answer validity score is calculated based on the number of retrieved documents. Another example is the work of Ko et al. [2007], who formalize the task of answer validation as the estimation of the probability $P(correct(A_i)|Q, A_1, ..., A_n)$ of a candidate answer A_i being correct given the posed question Q and the similarity between A_i and the other candidate answers. These authors use logistic regression and four types of features (knowledge-based, data-driven, string distances and synonyms) and report more than 100% of average improvement in results to 1760 questions collected from TREC, when compared to the baseline where the top scored answer is retrieved.

The importance of answer validation in a QA system motivated the creation of the Answer Validation Exercise (AVE) [Peñas et al., 2008] in CLEF. The AVE exhorts systems to automatically verify the correctness of an answer to a question, given the snippet that supports it. Most participating systems follow an approach based on textual entailment, where an hypothesis is built with the question and answer, and afterwards tested against the snippet.[2] To decide about the correctness of the answer, the competing systems use a large variety of approaches that can be roughly split into two categories:

[2]In 2006, the hypothesis was given instead of the pair question/answer. In the following years, each system was given a triple (question, answer, snippet).

- based on lexico-syntactic information – relying on the analysis of syntactic dependencies and/or overlapping of lexical or syntactic information between the text and the hypothesis. For instance, Iftene and Balahur [2009] compute a fitness value based on the matching between the named entities on the hypothesis and the entities of the supporting text.
- based on machine learning – using a training corpus and a set of features, such as the Levenshtein distance or the size of the longest common subsequence between the hypothesis and the supporting snippet [Téllez-Valero et al., 2008; Kozareva et al., 2006].

Re-ranking of candidate answers: A second group of techniques to answer selection aims at ranking candidate answers. This is done by attributing to each candidate a score that depends on its relevance to the given question. The best scored candidate answer(s) are ranked higher in the list of candidate answers and considered more likely to be the correct answer(s) to the question.

Thes score are calculated by taking into account several features extracted from the posed question, the candidate answer and, eventually, the sentence where it was found. Examples of features are the frequency of the candidate answer, the information overlap between the candidate answer and the question, and the rank of the sentence in the total number of relevant sentences. Some systems, like [Moldovan et al., 2000], heuristically combine these features, attributing a manually tuned weight to each of them. Other systems adopt machine learning-based strategies to automatically learn the feature weights. For instance, Ittycheriah et al. [2001] used a maximum entropy model to combine features and their weights; the features include query expansion features, focus features, named entity features and dependency relation features.

To conclude, we can say that selecting the final answer(s) boils down to distinguishing between correct and incorrect candidate answers. In this perspective, answer selection can be cast to a binary classification problem, where the system classifies each candidate answer as positive (correct) or negative (incorrect). If answer selection is seen as a filtering task,

the candidate answers that do not agree with the constraints are the negative examples; in candidate answer ranking, the classification depends on the answers' score against a threshold value: if the score is greater than the threshold, the candidate answer is a positive example, classified as correct; in candidate answer validation, the positive examples are the candidate answers that can be verified and justified by the system.

4.1.2 Relating answers

An approach to final answer selection in QA relies on choosing the most frequent candidate answer as final. This common, simple and successful approach is based on two important assumptions:

1. many instances of the correct answer exist in the information sources (although they can be stated in different formats); and,
2. candidate answers are unrelated and, therefore, competitors.

The work of Dalmas and Webber [2007] changed this perspective: they used relations between the answers to deal with the possible lack of redundancy of the information sources. In particular, they detected equivalences and inclusions between candidate answers to *where* questions, and organised the candidate answers in graphs. The computation of several features – like the semantic field, specificity and redundancy – allowed to rank the candidates.

Relating candidate answers can also be applied with the goal of comparing and integrating information chunks originated from different and heterogeneous sources. In this context, normalization is a step that typically proceeds the detection of relations. For instance, Moriceau [2005, 2006] compares and integrates answers of categories DATE and NUMBER, and Chu-Carroll et al. [2003] uses named entity normalization on the candidates retrieved from two answering agents.

Buchholz and Daelemans [2001] also considered relations in the context of the presentation of *complex* answers (answers composed of two or more simple answers). The authors identified

several situations where system's answer can be built by combining candidate answers through the relations they hold. For instance, given that *Sirius* and *Dog Star* refer to the same entity, to answer the question *What is the brightest stater?* the system should choose and show the user a supporting sentence that contains both. In fact, Buchholz and Daelemans [2001] reported that often the success of the system can pass by the recognition of the relations between candidate answers: for instance, if the answer *cars* is considered a more useful answer than *the pepper grinder which the company patented last century* to the question *What does the Peugeot company manufacture?*, the relations would allow a system to output a more appropriate answer to the question. The authors mentioned simple strategies that benefit from the existence of the relations, like, for example, selecting as final answer the one that includes most of the others.

4.1.3 A typology of relations

Webber et al. [2003] originally introduced a set of four relations between answers, which were later presented by Moriceau [2005].[3] Both consider the relations between correct answers, assuming a preceding filtering phase where the incorrect ones are discarded. On the contrary, and as we will see, in our work we deal with relations between (correct and incorrect) candidate answers and discuss how they influence the selection of the correct answer.

According to Webber et al. [2003] and Moriceau [2007], relations between answers can be of:

Equivalence – if answers are consistent and entail mutually. Equivalence exists between lexicographically different answers that represent the same entity, namely:

- answers with notational variations. For instance, *Oct. 14, 1947* and *14th October, 1947* are equivalent answers for *When did the test pilot Chuck Yeager break the sound barrier?*;

[3] In her Ph.D. thesis, Moriceau [2007] also refers to the relation of complementarity. The author introduces this relation as requiring the implementation of deeper inference mechanisms. For example, there is a relation of complementary between the candidate answers *a litter of water weighs 1 kg* and *for oil, the relative density of water is equal to 0.9* to the question *What is the weight of 30 litters of oil?*.

- answers that rephrase others, like synonyms or paraphrases. For example, the question *How did Jack Unterweger die?* can be answered with *committed suicide* or *killed himself*.

Inclusion – if answers are consistent and differ in specificity, one entailing the other. Inclusion occurs between two candidates that represent different entities, in which one includes or subsumes the other, through:

- hypernymy, that is, the answers are in a *is-a* relation. For example, *What animal is Kermit?* can be answered with *frog* or *amphibian*;
- meronymy, that is, the answers are in a *part-of* relation. For example, *Where did Ayrton Senna have the accident that caused his death?*, in which *Imola*, *Italy*, and *Europe* are possible answers;
- membership, that is, the answers are in a *instance-of* relation. For example, *Edvard Munch* is a member of *a Norwegian Symbolist painter*, both possible answers to *Who painted the "Scream"?*.

Aggregation – if answers are consistent, but not mutually entailing. In aggregation, all candidate answers are potentially correct and can be integrated in the form of a conjunction. For example, the question *What is Kermit?* can be answered with *frog* or *muppet*, or a conjunction of both: *frog and muppet*;

Alternative – if answers are not consistent or not mutually entailing. In the case of questions with unique answers, only one can be correct. For example, the question *How is the Pope?* can be answered with *ill* or *healthy*, but not with both. In the case of questions with multiple answers, all the alternatives may be distinct answers. For example, *twenty-eight* and *twenty-nine* are alternative answers to *How many days are in February?*.

Table 4.1 shows the existing relations between candidate answers, as presented by Dalmas and Webber [2007]. A parallel is also made with the work by Buchholz and Daelemans [2001], who focused on relations in the context of the presentation of complex answers (answers composed of two or more simple answers).

Moriceau [2005]	Webber et al. [2003]	Buchholz and Daelemans [2001]
Equivalence	Answers determined to be equivalent (mutually entailing)	Different measures Different designations Time dependency
Inclusion	Answers that differ in specificity (one-way entailing)	Granularity
Aggregation	Answers that are mutually consistent but not entailing	Collective answers
Alternative	Answers that are inconsistent	Many answers Ambiguity in the question Different beliefs

Table 4.1: Relations between candidate answers.

4.2 Final Answer Selection based on Semantic Relations

The simplest approach to final answer selection in QA relies on the frequency of occurrence of each candidate answer and considers the most frequent candidate as the final answer to the posed question. JUST.ASK also follows this type of redundancy-based approach, with a component dedicated to final answer selection that is mainly dependent on the number of times an answer was found in the relevant passages retrieved to a question (details can be found in Section 2.2.3.2).

In this section we study how the view of candidate answers as allies contribute to the task of final answer selection, by extending a pure redundancy-based approach with the information about the semantic relations existing between candidate answers. In order to do so, we part from the extracted candidate answers and follow the next four different sequential steps:

1. Candidate answers are normalized to a canonical format, which allows a better comparison between them: DATE answers are set to the form `D01 M01 Y1900` and numbers (written in alphabetic and/or numeric characters) are reduced to their respective numeric version with one decimal place (*1 hundred* is converted to *100.0*);

2. Every two answers are compared (in a case insensitive comparison). If they are equal we assume them to be the same entity and the score of each answer is increased by one. Here, the score of each answer is its frequency of occurrence;

3. The relations that exist between every pair of candidate answers are established. Again, we compare every pair of candidate answers and detect if a relation exists. Their scores are updated depending on their relations with other candidates;

4. The list of final answers is ordered by decreasing score. If two answers have equal score, the one that holds more relations is preferred over the other.

Note that the first two steps refer to the normalization and aggregation steps in JUST.ASK's architecture, respectively.

Regarding the relations, we use the typology presented by Moriceau [2005] and focus on equivalence and inclusion. In the following, we describe the used techniques for detecting the relations between two candidate answers and how they interfere with the redundancy-based approach to select the final answer in a group of candidate answers.

4.2.1 Detecting equivalence and inclusion

One of the main concerns in redundancy-based QA is to recognize equivalence between answers despite of their surface form, for instance, that *July 3rd, 1983* is equal to *07-03-1983*. As previously mentioned, this notion of equivalence goes from notational variations or reference resolution to the more complex concept of paraphrase, and it is an open issue in QA research. The score attributed to a certain answer is tightly coupled with its frequency and of its equivalents, and ignores other types of relations. However, it should be clear that, for instance, in the question *Where did Ayrton Senna have the accident that caused his death?*, the existence of the candidate answer *Imola* should boost the score of the candidate *Italy*, since they are connected though a relation of meronymy.

Depending on their semantics, we use different techniques to recognize if two answers are related by equivalence or inclusion, namely:

- We manually created regular expressions to detect equivalent candidate answers that refer to persons names. For instance, if a candidate A_1 is a string of alphabetic characters, we test if candidate A_2 matches the regular expression: "`(Mrs?\.?)|(Dr\.)|(Mister)|(Madame)` A_1". To cope with misspelled answers, we calculate the Levenshtein distance between every two answers and, if it is lower that a certain threshold, we assume the answers to be equivalent.

- In numeric answers, to deal with variation in numeric values, we assume that if the two answers differ by less than a threshold, then they are equivalent. Inclusion is detected with recourse to rules that test if a number is contained in an interval: for instance, *78.5* is included in *over 78.0*.

- When it comes to dates, we recognize inclusion by testing if one answer string contains the other. For instance, *1 January 2011* includes *1 January*, *January 2011* and *2011*.

- Regarding locations, we use a set of regular expressions to detect equivalence and inclusion. For example, *Republic of X* is equivalent to X and *University of Y* is included in Y.

- For other types of answer, we benefit from the knowledge present in WordNet. For each pair of lemmatized candidate answers, we query this lexical database for their most common sense. Afterwards, we navigate the tree of hypernyms (and meronyms) of each candidate's sense and check whether the other candidate is its ancestor. If so, a relation of inclusion exists. For example, this allows us to detect the relation of inclusion between candidate answers (*animal*, *sheep*) and (*body*, *arm*), since in WordNet the first argument of each pair is an ancestor of the respective second argument: a sheep *is-a* animal (hypermyny) and arm *is-part-of* body (meronymy). Moreover, we assume that equal lemmatized answers are equivalent.

After detecting the relations, one can built a directed graph of candidate answers, in which the answers are the nodes and the relations between them the edges. This approach was explored by Dalmas and Webber [2007], however their graph also contained information

to represent entities extracted from the question. Our representation of candidate answers is independent from the input question, since we only take into account the answers. Figure 4.1 depicts the graph of candidate answers, achieved after the detection of equivalence and inclusion relations. Straight lines represent relations of equivalence and arrows represent relations of inclusion, in which the answer at the start of the arrow includes the answer at the end of the arrow.[4]

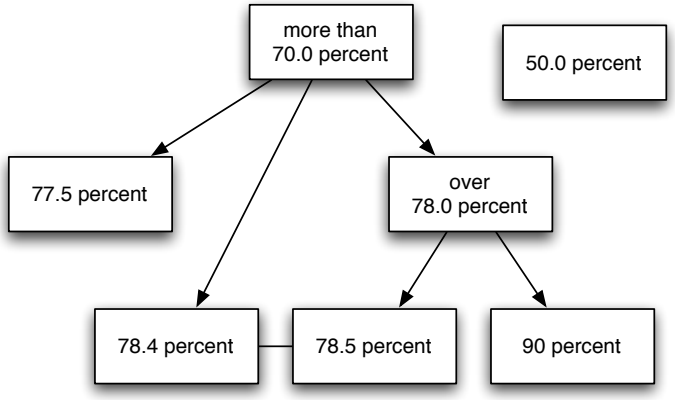


Figure 4.1: Relations between numeric candidate answers.

4.2.2 Selecting the final answer

In the answer selection approach based uniquely on frequency, candidates are scored independently and, for that reason, considered as autonomous. Here, the more popular an answer is, the more chances it has to be chosen as the final answer.

In the extended answer selection approach, candidate answers are scored according to the relations they hold with others, besides their frequency. The underlying assumption is that the correctness of an answer is influenced by the presence of the other answers with which it is related. Here, and recalling the graph representation of the candidate answers, the more connections an answer has, the more chances to be chosen as the final answer. Being so, we implemented a scoring system, which updates the score of an answer A according to its type and frequency, to the number of answers equivalent to A, to the number of answers which

[4]For simplicity reasons, we omitted transitive inclusions.

include A and to the number of answers that are included by A. The strategy boils down to navigating the graph of candidate answers and adding to the score of each node a weight that depends on the number and type of edges that leave from and arrive to it. Like in the frequency based approach, a list is built from the graph, ordered by decreasing value of score, and the topmost is chosen as final.

4.3 Evaluation

4.3.1 Experiments

The focus of our experiments is on answers to questions of categories ENTITY, LOCATION, NUMERIC:COUNT, NUMERIC:DATE and HUMAN:INDIVIDUAL. Thus, in all of the following, the questions were first classified according to Li and Roth's two-layer question type taxonomy [Li and Roth, 2002], using JUST.ASK's question classifier. Afterwards, we discarded all answers of coarse-grained categories ABBREVIATION and DESCRIPTION of the unwanted fine-grained categories (including, HUMAN:DESCRIPTION, HUMAN:GROUP and HUMAN:TITLE).

We conducted two main experiments, which allow us to compare the impact of a final answer selection based on relations when the extracted answers are originated from a single source or multiple sources (systems). The details follow:

Experiment 1 – Multi-stream question-answering In a first experiment we tested our approach in a setting where a large number of candidate answers to factoid questions exist, as well as their respective judgements as correct or incorrect.[5] To deal with the difficulties in gathering such corpus from one unique system, we again decided to use the freely available data from the TREC QA tracks. It contains not only the test questions, but also the answers given by the competing systems and judged as correct or incorrect by the human assessors of TREC. Being so, all our answers are originated in actual QA systems that used the same information sources to solve every question and their

[5] In [Mendes and Coheur, 2011] we show the results of a similar experiment.

evaluation is trustworthy. We consider this to be a reliable and consistent corpus, that allows to mimic the behavior of a real system, in line with a recent trend in QA – multi-stream QA – in which the output of several and different systems (streams) are combined to improve the answer accuracy [Téllez-Valero et al., 2010].

In this experiment, we tested two different settings where we varied the number of questions of the evaluation corpus: on the first one, we use all the questions from the TREC competition, years 2002 to 2007; on the second setting, we use corpus employed in JUST.ASK's evaluation described in Section 2.5.2.[6]. Moreover, and since we aim at evaluating if the approach is able to better position the correct answer in the list of final answers, we only use those questions for which JUST.ASK extracts at least one correct answer. This second setting allow us to make a comparison between the achieved results in multi- and single-stream QA, specifically those achieved by JUST.ASK. In the following, we detail these two settings:

Setting a) We dealt with a total of 46,068 answers, from which 10,198 (22.14%) were judged as correct and 34,756 (77.86%) as incorrect.[7] These answers belong to 1,642 different questions collected from the TREC QA tracks of the years of 2002 to 2007, which had at least one correct answer. Roughly, there is an average of 28 candidate answers per question, and only 1 in every 5 is correct. Details of the corpus according to the category are presented in Table 4.2.[8]

Setting b) We used the questions from TREC-QA_2002-2007 corpus (for which JUST.ASK extracted at least one correct answer). The answers are those returned by the competing systems of TREC.

We dealt with a total of 19,053 answers, from which 5,872 (30,82%) were judged as

[6]Recall that this corpus is composed of a subset of the questions of the TREC competition, years 2002 to 2007. Particularly, this corpus only contains factoid, non-anaphoric questions; their answers are short-sized, returned by the competing systems of the competition and judged as correct or unsupported. Section 2.3.2 further details the creation of this corpus.

[7]In order to have a larger amount of correct answers, inexact and unsupported answers are also considered as being correct.

[8]Although in Li and Roth's taxonomy the fine-grained category DATE belongs to the coarse-grained category NUMERIC, we separate them to differentiate the results achieved for dates or numbers. Being so, where we have NUMERIC, it should be understood as all questions/answers classified as NUMERIC except NUMERIC:DATE

Category	# Questions All	# Answers Correct	Incorrect	All
HUMAN:INDIVIDUAL	371	2,499 (23.19%)	8,278	10,777
ENTITY	270	1,175 (15.82%)	6,414	7,589
LOCATION	400	3,675 (31.57%)	7,964	11,639
NUMERIC:COUNT	204	1,114 (19.82%)	4,507	5,621
NUMERIC:DATE	397	2,849 (27.28%)	7,593	10,442
Total	1,642	10,198 (22.14%)	34,756	46,068

Table 4.2: Details of the corpus used in Experiment 1, setting a).

correct[9] and 13,181 (69.18%) as incorrect. These answers belong to 693 different questions. Roughly, there is an average of 27 candidate answers per question, and roughly 1 in every 3 answers is correct. Details of the corpus according to the category are presented in Table 4.3.

Category	# Questions All	# Answers Correct	Incorrect	All
HUMAN:INDIVIDUAL	144	1,030 (24.95%)	3,128	4,158
ENTITY	29	166 (21.25%)	615	781
LOCATION	235	2,441 (36.50%)	4,246	6,687
NUMERIC:COUNT	72	492 (26.28%)	1,380	1,872
NUMERIC:DATE	213	1,743 (31.38%)	3,812	5,555
Total	693	5,872 (30,82%)	13,181	19,053

Table 4.3: Details of the corpus used in Experiment 1, setting b).

Experiment 2 – Single-stream question-answering In a second experiment, we used the questions from the TREC-QA_2002-2007 corpus and the candidate answers extracted by JUST.ASK (again, we only use those questions for which the system extracts at least one correct candidate answer).

Since the TREC-QA_2002-2007 corpus does not contain all the variations of the correct answers, we do not have the actual judgement to every answer extracted by JUST.ASK. Our way to circumvent this issue was to assume that all answers that are normalized to the same format as of a correct answer are correct. In this way, we dealt with a total of 76,708 answers, from which 34.64% are correct. The ratio of correct answers per category is somewhat similar to that in the previous experiments, the exceptions are

[9]Unsupported answers are considered correct. Inexact are considered incorrect.

categories ENTITY and NUMERIC:DATE, where the number of correct answers is around 14% higher. Roughly, there is an average of 110 candidate answers per question, and 1 in every 3 is correct. Details of this corpus are presented in Table 4.4.

	# Questions	# Answers		
Category	All	Correct	Incorrect	All
HUMAN:INDIVIDUAL	144	4,250 (25.83%)	12,204	16,454
ENTITY	29	1,153 (35.89%)	2,060	3,213
LOCATION	235	14,378 (36.52%)	24,987	39,365
NUMERIC:COUNT	72	1,579 (26.60%)	4,367	5,936
NUMERIC:DATE	213	5,219 (44.45%)	6,521	11,740
Total	693	26,579 (34.64%)	50,136	76,708

Table 4.4: Details of the corpus used in Experiment 2.

Scoring

In all the experiments, the answers are selected by decreasing order of score. In this section, the results of the conducted experiments are shown according to the following:

Frequency The baseline consists in selecting the most frequent candidate answer, without any other processing, and the respective results are those in column Frequency. Therefore, the score of an answer a_i equals its frequency;

Frequency+Normalization The results of the impact of answer normalization in the baseline are shown in column Frequency+Normalization. Therefore, the score of an answer a_i equals its frequency after normalization;

Frequency+Normalization+Relations In column Frequency+Normalization+Relations we show the results of selecting the candidate answers based on the number of relations established with the other candidates. Here, the score of an answer a_i equals its frequency after normalization. If two answers have the same score, they are ordered according to the number of relations.

Frequency+Normalization+Relations+Scoring The performance of the answer selection approach based on semantic relations (which involves candidate answer normal-

ization, scoring and selection based on the relations) is presented in column Frequency+Normalization+Relations+Scoring. Here, the score of an answer a_i equals its frequency after normalization plus the score associated with the relations it holds with others. If two answers have the same score, they are ordered according to the number of relations.

In case there is still a tie between two answers, the selection is based on alphabetical order.

For clarity, consider as an example the following candidate answers: *July 1983*, *July*, *07/1983*, *July 3rd,1983*, *1983*, *August*, *August 1983*, *August 1983*. In Table 4.5 they are ranked by decreasing score (which is also presented).

Frequency	Frequency +Normalization	Frequency +Normalization +Relations	Frequency +Normalization +Relations +Scoring
August 1983 ⇒ 2	*August 1983* ⇒ 2	*July 1983* ⇒ 2	*July 1983* ⇒ 4
07/1983 ⇒ 1	*July 1983* ⇒ 2	*August 1983* ⇒ 2	*July 3rd, 1983* ⇒ 4
1983 ⇒ 1	*1983* ⇒ 1	*1983* ⇒ 1	*August 1983* ⇒ 4
August ⇒ 1	*August* ⇒ 1	*July 3rd, 1983* ⇒ 1	*1983* ⇒ 1
July ⇒ 1	*July* ⇒ 1	*August* ⇒ 1	*July* ⇒ 1
July 1983 ⇒ 1	*July, 3rd 1983* ⇒ 1	*July* ⇒ 1	*August* ⇒ 1
July 3rd, 1983 ⇒ 1			
(a)	(b)	(c)	(d)

Table 4.5: Example of the final answer selection as presented in this section.

Regarding the notes in the last row of the table, note that:

(a) there are two candidate answers *August 1983*, therefore it will be scored 2. The score of the remaining others is 1;

(b) both *July 1983* and *07/1983* normalize to *M07 Y1983*, therefore they are aggregated with score 2. The score of the remaining is kept unchanged;

(c) *July 1983* is related to 3 answers (*1983*, *July* and *July 3rd, 1983*), whereas *August 1983* only to two (*1983* and *August*), therefore they will be swapped in the list (with *July 1983* ranking higher than *August 1983*). The same logic applies to the other candidate answers;

(d) Assuming that, in a relation of inclusion, the score of the candidate answer that is included is increased by one:

- the score of *July 3rd, 1983* is increased by 3, since it is included by *1983*, *July 1983* and *July*;
- the score of *July 1983* and *August 1983* are increased by 2 since they are included by *1983* and *July*, and *1983* and *1983*, respectively.

Furthermore, since *July 3rd, 1983* and *July 1983* are related to 3 other candidate answers, while *August 1983* to only 2, they are ranked higher than the latter candidate answer and according to their alphabetical order. Again, the same logic applies to the remaining candidate answers.

Finally, the scores to each relation were fixed and do not take into consideration the posed question; however, and for example, different questions might expect different answer granularities. In these experiments we report the upper bound of accuracy when using relations for the purpose of answer selection, considering that the system can identify the best values for the scores in order to maximize its MRR.

4.3.2 Results

4.3.2.1 Experiment 1 – Multi-stream question-answering

Setting a) Detailed results achieved in setting a) of experiment 1 – Multi-stream question-answering – are shown in Table 4.6.

The baseline accuracy is of 56.52% for the 1,642 questions. Results are pushed down mostly because of the performance in the categories ENTITY and NUMERIC:COUNT, both achieving accuracies at the first answer below 50%. We can see that the system fails in the answer selection task as the number of correct answers at position 3 is much higher than that of position 1.

	Frequency			**Frequency +Normalization**			**Frequency +Normalization +Relations**			**Frequency +Normalization +Relations +Scoring**		
	Acc	#Correct		Acc	#Correct		Acc	#Correct		Acc	#Correct	
Cat	@1	@1	@3	@1	@1	@3	@1	@1	@3	@1	@1	@3
H:IND	54.18%	201	280	54.18%	201	280	57.41%	213	291	59.78%	220	296
ENT	47.03%	127	175	47.03%	127	175	49.25%	133	188	49.25%	133	183
LOC	69.75%	279	347	69.75%	279	347	70.50%	282	351	73.50%	294	350
N:COU	42.16%	86	135	53.43%	109	165	54.41%	111	162	56.83%	113	164
N:DAT	59.19%	235	319	64.99%	258	327	66.25%	263	331	68.77%	273	331
Overall	56.52%	928	1,256	59.32%	974	1,294	61.08%	1,002	1,323	63.15%	1,033	1,324
Relative change					4.96%	3.03%		2.88%	2.16%		3.09%	0.08%

Table 4.6: Results achieved in experiment 1 – Multi-stream Question-Answering, setting a).

After normalizing the questions pertaining to categories NUMERIC:COUNT and NUMERIC:DATE (recall that the others are not normalized), the overall results increase nearly 3% (in absolute values) when compared to the baseline. A total of more 46 questions are now correct at rank 1. Considering the top 3 ranked answers, results improve and are again better than those achieved by the baseline: 38 more questions have correct answer within the top 3.

The selection of answers based on the number of relations established with other candidate answers allows an absolute increase of more than 1.5% in the accuracy@1 of the entire dataset, with 28 more questions being correctly put in the first position of the list of final answers. The highest improvements are seen in answers to questions of categories HUMAN:INDIVIDUAL, and ENTITY and NUMERIC:DATE, with an increase of accuracy@1 around 3% and 2%.

Finally, scoring candidate answers based on the relations results on an increase of around 7%, 4% and 2% (in absolute values), when compared to using only Frequency, Frequency+Normalization and Frequency+Normalization+Relations, respectively. The top answers are correct for a total of 1,033 questions; the difference is of 105 and 68 questions at top 1 and 3, comparing with the baseline. Results suggest that the approach that takes the semantic relations into account better groups the correct answers in the top positions of the list of scored candidate answers.

Regarding the number of relations between answers, we detected a total of 12,987 re-

lations (6,474 equivalences and 6,504 inclusions), which particularly benefitted categories HUMAN:INDIVIDUAL, LOCATION and NUMERIC:DATE. To assess the correctness of the detected relations, we collected and manually evaluated a random sample of 100 relations for each category. The results are shown in Table 4.7.

	All		**Equivalence**		**Inclusion**	
Cat	#Correct	Precision	#Correct (Total)	Precision	#Correct (Total)	Precision
H:IND	72	72.00%	72 (100)	72.00%	–	–
ENT	53	53.00%	8 (28)	28.57%	45 (72)	62.50%
LOC	93	93.00%	29 (29)	100.00%	64 (71)	90.14%
N:COU	92	92.00%	51 (57)	89.47%	41 (43)	95.35%
N:DAT	95	95.00%	–	–	95 (100)	95.00%
Overall	405	81.00%	160 (214)	74.77%	245 (286)	85.66%

Table 4.7: Evaluation results of the relations detected in setting a) of experiment 1 – Multi-stream question-answering.

The worst precision result was verified on category ENTITY. An explanation to this result is that many answers were snippets of text, often composed of several entities (or no entities at all), instead of one unique entity whose semantics agree with the question category. For example, *Racing Online Series*, *totally nascar*, *White House dealt* and *Regiment one of* were candidate answers to questions of category ENTITY. A similar issue occurred with category HUMAN:INDIVIDUAL, which yielded the second lowest results, as many candidates were not names of individuals. For example, *classic works*, *short form music video* or *Atlantic Ocean*. Given that our strategies to detect relations expect answers of a certain semantics, these situation led to relations being wrongly established between candidate answers. The results for the remaining categories were higher than 90%, leading to an overall precision result of 81.00%.

Setting b) Table 4.8 presents the detailed results obtained in setting b) of experiment 1 – Multi-stream question-answering.

The accuracy results of the baseline are of around 71%. This value increases to a maximum of 79.07%, achieved when normalization is applied and the relations are used to select and score the answers (a difference of more than 8% when compared to the baseline). The existence of the relations are responsible for more 51 correct answers at rank 1 and 34 at rank 3. These

	Frequency			**Frequency +Normalization**			**Frequency +Normalization +Relations**			**Frequency +Normalization +Relations +Scoring**		
	Acc	#Correct		Acc	#Correct		Acc	#Correct		Acc	#Correct	
Cat	@1	@1	@3	@1	@1	@3	@1	@1	@3	@1	@1	@3
H:IND	61.11%	88	115	61.11%	88	115	64.68%	93	118	68.75%	99	123
ENT	72.41%	21	24	72.41%	21	24	72.41%	21	27	79.31%	23	28
LOC	80.85%	190	222	80.85%	190	222	81.28%	191	224	87.50%	198	223
N:COU	61.11%	44	59	73.61%	53	69	75.00%	54	68	76.39%	55	68
N:DAT	69.01%	147	186	73.24%	156	194	75.59%	161	196	77.93%	166	198
Overall	70.70%	490	606	73.30%	508	624	75.04%	520	633	79.07%	541	640
Relative change					3.68%	2.97%		2.36%	1.44%		4.04%	1.11%

Table 4.8: Results achieved in experiment 1 – Multi-stream question-answering, setting b).

results are consistent with the ones achieved in the setting a) of this experiment.

We detected a total of 4,064 relations (2,146 equivalences and 1,918 inclusions), and their impact in the accuracy results was mostly seen in categories HUMAN:INDIVIDUAL and NUMERIC:DATE, which gained 21 and 19 correct questions at rank 1, respectively. Categories LOCATION and NUMERIC:COUNT also benefitted from the relations, however in a smaller amount (more 8 and 11 correct answers in each category, respectively).

4.3.2.2 Experiment 2 – Single-stream question-answering

Detailed results achieved in experiment 2 – Single-stream question-answering – are shown in Table 4.9.

Although the best accuracy is achieved when applying normalization and using the relations between answers (60.61%), the difference to the baseline is not that expressive as it was in the previous experiments: less than 3% improvement was seen in this experiment, representing 18 more correct questions at rank 1. The number of correct questions until rank 3 also increases by 22.

From the analysis of the table, we put in evidence three results, which mostly differed from previous experiments:

	Frequency			**Frequency +Normalization**			**Frequency +Normalization +Relations**			**Frequency +Normalization +Relations +Scoring**		
	Acc	#Correct		Acc	#Correct		Acc	#Correct		Acc	#Correct	
Cat	@1	@1	@3	@1	@1	@3	@1	@1	@3	@1	@1	@3
H:IND	50.0%	72	111	33.8%	72	111	49.31%	71	114	50.69%	73	116
ENT	72.41%	21	27	72.41%	21	27	75.86%	22	29	75.86%	22	29
LOC	63.83%	150	194	63.83%	150	194	64.26%	151	194	64.26%	151	195
N:COU	44.44%	32	52	50.00%	36	54	50.00%	36	55	51,39%	37	58
N:DAT	59.62%	127	169	61.03%	130	174	62.64%	133	178	64.31%	137	180
Overall	58.01%	402	553	59.02%	409	560	59.60%	413	570	60.61%	420	575
Relative change					1.74%	1.27%		1.79%	0.98%		1.69%	0.88%

Table 4.9: Results achieved in experiment 2 – Single-stream question-answering.

- The small increase in the accuracies achieved for categories LOCATION and HUMAN:INDIVIDUAL when using relations for selection and/or scoring. In previous experiments, these categories largely benefitted from the usage of relations. We believe that a possible reason for this is related with the enormous quantity of candidate answers, which hide the impact on the relations in the results. Moreover, and regarding category LOCATION, there was also a relatively small number of discovered relations between candidates: 418 relations for nearly 39,365 candidates (against previous experiments where 570 and 324 relations were discovered for 11,639 and 6,687 candidates, respectively).

- The highest increase in the accuracy reported for category NUMERIC:DATE, which consistently benefited from the relations throughout the experiments. Moreover, note that, in this category, only relations of inclusion were detected.

- The relations between the answers seemed to have more impact in the results when they were used for scoring, rather than for selecting between two equally scored candidate answers.

In this setting, we detected a total of 16,191 relations (8,594 equivalences and 7,576 inclusions). The results of the manual evaluation performed to the detected relations is shown in Table 4.10 (again, we collected a random sample of 100 relations for each category).

	All		Equivalence		Inclusion	
Cat	#Correct	Precision	#Correct (Total)	Precision	#Correct (Total)	Precision
H:IND	96	96.00%	96 (100)	96.00%	–	–
ENT	77	77.00%	9 (11)	81.82%	68 (89)	76.40%
LOC	80	80.00%	6 (8)	75.00%	74 (92)	80.43%
N:COU	96	96.00%	38 (38)	100.00%	58 (62)	93.55%
N:DAT	98	98.00%	–	–	98 (100)	98.00%
Overall	447	89.40%	149 (157)	94.90%	298 (343)	86.88%

Table 4.10: Evaluation results of the relations detected in experiment 2 – Single-stream question-answering.

Like in the previous evaluation, the category ENTITY is the one that achieves the lowest results, although significantly higher than before: 77.00% *vs.* 53.00%. Here in this evaluation, however, the second lowest result is that of category LOCATION (instead of category HUMAN:INDIVIDUAL). Again, the issue was mostly because several relations were established between answers that were not locations, *e.g.*:, `includes(`*Algeria*`,`*Flag of Algeria*`)` and `includes(`*Britain*`,`*Emperor of Britain*`)`, due to the match of the implemented regular expressions. In the remaining categories the results were above 90% of precision and the overall results also higher than those achieved in the previous experiment.

4.3.2.3 Other results

In situations where a very large text corpus is available (ideally the Web), the approach to QA based on the redundancy of the information sources is appealing: it achieves good results while relying mostly on counting the number of occurrences of each candidate answer and on the fact that the correct answer is typically the most frequent. In all previous experiments, we assumed the redundancy of the information sources, since we allowed each candidate answer to be repeated multiple times (like as if it was present several times in the information sources). However, this is not often the case, specially when using closed corpora of small dimensions as the source of information. In this experiment, we simulate the behavior of a system in a situation of lower redundancy. We did so by considering that all the candidate answers are unique, that is, there is only one instance of each candidate answer. In this case, the selection of the answers is not dependent on their frequency, but on the relations with other

answers. If all candidate answers are unique and have the same frequency (that is, 1), and if no other information is available, a QA system would probably choose randomly amongst all answers.[10]

The results achieved when using the approach to answer selection based on semantic relations, while disregarding the frequency of the candidate answers, are the following:

Experiment 1, setting a) an accuracy@1 of 42.14% was achieved, with 692 questions correctly answered at the top 1 rank. 954 questions are correctly answered at top 3.

Experiment 1, setting b) an accuracy@1 of 53.54% was achieved, with 371 questions correctly answered at the top 1 rank. 497 questions are correctly answered at top 3.

Experiment 2 an accuracy@1 of 22.08% was achieved, with 153 questions correctly answered at the top 1 rank. 268 questions are correctly answered at top 3.

Table 4.11 further details the results according to the selection method previously described. Note that the difference to previous experiments is that here every candidate answer is unique and its score is set to one.

	Frequency (= 1)			**Frequency** (= 1) **+Normalization**			**Frequency** (= 1) **+Normalization +Relations**			**Frequency** (= 1) **+Normalization +Relations +Scoring**		
	Acc	#Correct		Acc	#Correct		Acc	#Correct		Acc	#Correct	
Experiment	@1	@1	@3	@1	@1	@3	@1	@1	@3	@1	@1	@3
1 – a)	9.07%	149	483	23.45%	385	602	40.68%	668	1,006	42.14%	692	954
1 – b)	10.68%	74	243	29.44%	204	362	47.91%	332	487	53.54%	371	497
2	1.88%	13	63	10.97%	76	81	15.73%	109	209	22.08%	153	268

Table 4.11: Answer selection results when every candidate answer is unique.

As it can be seen, the use of the relations is determinant for the purpose of answer selection in a scenario of low redundancy, which we simulated by assuming that all the candidate

[10] In order to allow comparisons between the experiments, instead of randomly ranking the answers, we used their alphabetical order.

answers are unique. Regarding the results when the answers are ordered uniquely according to their alphabetical order (Frequency (= 1)), we can see that experiment 2 has much smaller values than those of experiments 1,a) and 1,b), which did not happen previously: we attribute this to the much larger amount of unique candidate answers of the corpus used in experiment 2. The normalization applied to the candidates allowed the accuracy to be largely boosted in every experiment, but the best overall results were attained when using the relations.

The achieved results show that, although answer redundancy is a good measure of the correctness of an answer, when this property is not available, using normalization with the semantic relations between answers seems to be a good substitute.

4.3.3 Influence in Just.Ask

Recall from Chapter 2 that answer selection in JUST.ASK comprises the following sequential steps: normalization, aggregation, clustering, filtering and selection. In order to accommodate the new approach to answer selection based on semantic relations, JUST.ASK was extended in the following way:

- before clustering, the relations between the candidate answers are detected;
- the answers' scores are updated according to the relations they hold with others. This step also occurs before clustering.
- when clustering, we assume a distance between equivalent answers of 0.0. Therefore, equivalent answers will belong to the same cluster.

Moreover, we did not perform any clustering on the answers to questions of category NUMERIC:DATE, as we have noted that applying the Levenshtein distance will make the clusters to include totally diverse dates.

Results achieved in the previous single-stream QA scenario let anticipate a comparatively small increase of performance. In fact, if one considers all the answers contained in the

returned clusters (as we did on the evaluation of the baseline in Section 2.5.2.4), we see that there is a improvement of 4 and 10 more correct answers (clusters) at rank 1 and 3, respectively.

Similar results are seen if we only consider the representative answer of each cluster, *i.e.*, the answer that the system *selects* as final and will return to the user. In this case, there is an increase of 4 correct answers at rank one (384 *vs.* 388 correct answers) and 9 at rank 3 (606 *vs.* 615 correct answers).

4.4 Discussion

Our approach to final answer selection is inspired on the research by Dalmas and Webber [2007]. However, and contrary to these authors, we show the importance of the semantic relations to a large spectrum of questions, not only *where* questions. Another difference to the previous work is that our candidate answers are the output of real-world QA systems, instead of being synthetically generated. In our work, we consider the relations as a way to improve the results already achieved by redundancy, but we also report the results of using relations to final answer selection when the candidate answers are not redundant.

We presented the performance of the answer selection approach based on semantic relations and, as it could be seen, results were different when using the answers retrieved from several systems (in a multi-stream question-answering scenario) and when using the answers retrieved by a unique system (in a single-stream question answering scenario). We believe that the provenance of the answers played a major role in the achieved results. Indeed, different systems resort to different strategies (to question interpretation, passage retrieval and answer extraction) and knowledge sources, which eventually lead to more diversity in the candidate answers. Moreover, the returned systems' answers are already the result of a filtering process, and are those that the systems have most confidence in being correct. This approach is feasible under the circumstances where correct answers are similar enough (but not equal) so that many relations can be established between them, and incorrect answers are diverse

enough so that no relation is recognized between them.

Regardless of the scenario and of the overall impact in the achieved results, the detection and usage of the semantic relations between candidate answers for the purpose of answer selection improved the accuracy achieved at ranks 1 and 3, suggesting the applicably of the approach. Moreover, many other techniques could be used to relation detection, which we did not explore (for example, based on georeference for answers to questions of category LOCATION) that can improve the results. The improvement of the current techniques, as well as the implementation of others (for example, based on paraphrase recognition) is, therefore, a future work direction to be considered in JUST.ASK.

Another point that is worth mentioning is related with the score attributed to each answer, which depends on the relations the answer has with others. In our approach, in every experiment, we have fixed a score attributed to each relation and depending on the question's category. However, the scores ought to be adjusted according to finer-grained information, for instance, to the posed question itself. Take the following two questions as examples: *When did Bob Marley die?* and *What day did Neil Armstrong land on the moon?*, which, in JUST.ASK, are classified as NUMERIC:DATE. While the former can be correctly answered with a year or even a decade, the latter requires a more specific temporal point (a day). Therefore, in the first one, more weight should be given to answers that include other candidate answers, whereas in the second more weight should be given to answers that are included by others. In this work, we only took in consideration the information from the candidate answers, but accounting for other clues in the question to improve the selection of the best answer is certainly another direction of future work.

4.5 Summary

We presented an approach to answer selection in QA that takes into account not only the candidate answers' frequency, but also the relations they hold with other candidate answers. Using a limited set of heuristics, encoded mostly in the form of regular expressions, as well

as linguistic knowledge from WordNet, we built a graph which we traverse to update the score of every answer. With this approach that uses mostly information recovered from the answer, we could boost the performance of the baseline in a multi-stream QA setting in more than 7%. Results were inferior in a single-stream QA, where we could only reach nearly 3% improvement. We presented a detailed evaluation and we discussed the impact of frequency, normalization and the semantics relations for the purpose of ranking candidates and selecting the final answer.

5 From Answered Questions to Question Answering

In a typical interaction with a QA system, a question is posed and its answer is returned. Usually, after being presented to the user, the system's answers are discarded from further processing. However, there is much information conveyed by the correct answer to a question that is simply lost in every interaction. For example, knowing that *Dante* correctly answers the question *Who wrote the Divine Comedy?* and that it can be found in the sentence *Dante has written The Divine Comedy* might be important to locate the correct answer to the *similar* question *Who painted The Birth of Venus?*.

JUST.ASK answers questions using several strategies that recognize and pinpoint named entities from text, in combination with a pattern-based approach to candidate answer extraction, described in Chapter 3. In this chapter we propose the application of this approach in a setting where the system learns to answer new questions based on previous successful interactions.

The present chapter is dedicated to the process of learning to answer new questions based on previous successful interactions, which was implemented in JUST.ASK. It starts with a revision of related work and, afterwards, our approach to learn to answer from answered questions is described. The evaluation and its results are reported and the chapter ends with the usual brief summary.

5.1 Related Work

Many works in the literature use questions and their correct answers in strategies for QA (several of them take advantage of the datasets of questions and answers built in the context

of evaluation fora, like the TREC or the CLEF). An example is our approach to candidate answer extraction that uses pairs of questions and their answers to learn extraction patterns; we have pointed to other approaches with the same goal in Section 3.1.

The use of Q/A pairs as training instances to (semi-)supervised learning machinery in QA is however not limited to the task of building patterns to candidate answer extraction. Sun et al. [2006] uses Q/A pairs to learn classifiers that identify the correct answers for a question in a sentence. The first classifies a sentence as containing (or not) a correct answer to the posed question; if the sentence is classified positively, the second classifies each word as correct (or not). Moschitti and Quarteroni [2011] focus on the answer selection to definition questions, and base their work on the cross-pair similarity model [Zanzotto and Moschitti, 2006] that learns rewrite rules between two entailment pairs (*Text T*, *Hypothesis H*). They use questions and their answers as training instances and study the improvements achieved when using generalizations to syntactic/semantic structures and applying sequence/tree kernel technology in a SVM. Lita and Carbonell [2004] represent questions as points in a multidimensional space and group them in clusters according to their similarity, based on the idea that similar questions are solved by similar strategies. Different models are learned from each cluster that serve three different purposes: 1) estimate a distribution of the question's semantic category; 2) include cluster-specific content in the queries submitted to the document retrieval module; and, 3) identify if an answer is present in a text snippet. When a new question is posed, it is represented in the same space and the models of the clusters in its neighborhood are used.

As we will show in the remainder of this chapter, our pattern-based approach to candidate answer extraction allows Just.Ask to learn to answer new questions based on previous successful interactions. For that, the system relies on the user feedback about the correctness of the returned answer(s). When it comes to systems that use past *interactions* to answer new questions, the approach of Harabagiu et al. [2001] is a rare example described in the literature: by using a caching mechanism, answers to previous similar questions are reused, which avoids triggering the entire QA process. The similarity is measured in terms of the number of (lexico and semantic) matches between content words of equal morphological category in

both (current and cached) questions. Other works exist along the line of reusing answers from past questions in recent questions, specially applied to Community Question-Answering sites.[1] Here, the goal is to reduce to amount of unanswered questions and the main challenge is typically to identify the past question(s) that is(are) the most similar to the new question, the one(s) that convey the same information need. For example, Shtok et al. [2012] use a two-stage approach that, besides choosing the most appropriate past question through a ranking mechanism, also identifies the best answer to the new question from the list of answers to the top-ranked past questions. Finally, it should be mentioned that, although the direct user feedback is rarely used in traditional QA, its application is explicit in this type of sites, where answers are ranked based on several features, including the number of votes attributed by the community members.

In Chapter 3 we presented several strategies to candidate answer extraction that use patterns built from Q/A pairs, including ours. In this chapter we will describe how we use our strategy to allow the system to learn from past interactions, making JUST.ASK, to the best of our knowledge, the first system that uses the correct answers to past questions to learn to answer future questions which, contrary to the approaches we have referred to in this section, do not necessarily convey the same information need.

5.2 Learning to Answer from Answered Questions

This section is dedicated to describe our method that uses previous information (questions and their correct answers) to answer new questions. Moreover, we show how JUST.ASK's architecture was modified in order to allow it to learn from past interactions.

[1] In these sites, questions and answers are given by humans. Examples of such sites are Yahoo! Answers (`http://answers.yahoo.com/`), StackOverflow (`http://stackoverflow.com/`) or Quora (`http://www.quora.com/`).

5.2.1 Iterative learning to answer

In Chapter 3 we introduced our pattern-based approach to candidate answer extraction. The assumption behind this approach is that, if two questions are similar in the sense that they share some properties (the syntactic structure and semantic category), then there is a set of patterns that can be used to extract answers to both of them. Given that the patterns are learned from pairs of questions and their correct answers, we are able to use the patterns learned from previous successful interactions (where the system correctly solved the posed questions) to extract answers to unseen *similar* questions.

When a new question is posed, its classification and syntactic analysis (performed in the Question Processing stage) allows to retrieve from the knowledge base all the applicable patterns. Relevant passages are returned in the Passage Retrieval stage and, in the Answer Extraction stage, there is a pattern/sentence unification to extract the candidate answers: every sentence in the passages is analyzed and its parse tree is explored in a top-down, left-to-right, depth-first search, unifying the obtained sentence tree components with the lexico-syntactic information in the patterns.

Since the pattern-learning algorithm receives a question and its answer as input, if the system is able to verify the correctness of its answers, the process to learn new patterns is straightforward: the input question is the question posed by the user, while the input answer is the system's retrieved correct answer. Thus, if positive feedback is given, JUST.ASK learns new patterns with previously (and correctly) answered questions.

5.2.2 Just.Ask's new architecture for answering questions

JUST.ASK's architecture was modified in order to accommodate the process of iterative learning. Figure 5.1 depicts the new architecture. The block labelled with number 2 depicts the steps in JUST.ASK dedicated to answering questions, while the block labelled with number 1 shows the steps in JUST.ASK dedicated to learning patterns. The components responsible for the Question Processing and Passage Retrieval steps are shared by both blocks, but for

the sake of readability of the figure, we have drawn a separate instance of each component in each block.

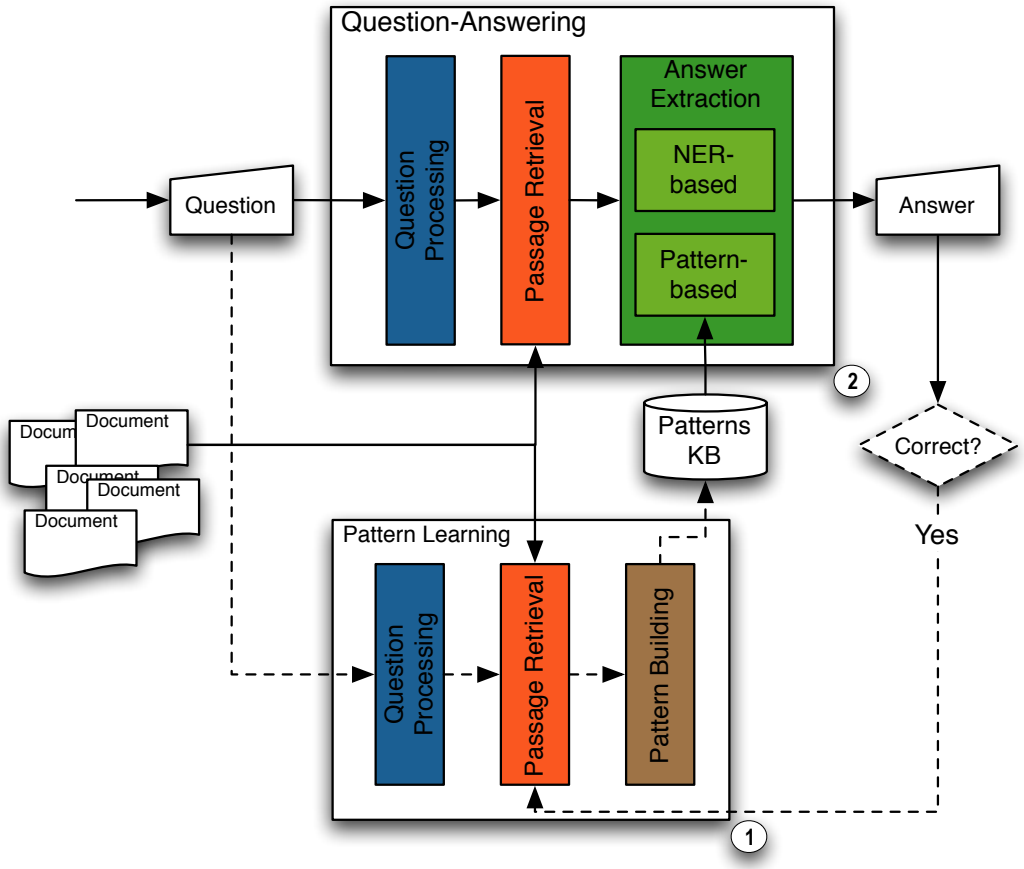


Figure 5.1: Learning from answered questions: the new architecture of JUST.ASK.

The data flow of JUST.ASK now comprises the following steps:

1. A new question is posed to JUST.ASK;
2. **Question Processing** – Interprets the question;
3. **Passage Retrieval** – Retrieves a set of relevant passages/documents for the question, from the information sources;
4. **Pattern Selection** – Retrieves a set of patterns for the question from the knowledge base of patterns.

5. **Answer Extraction – Candidate Answer Extraction**

   (a) **Pattern Matching** – If the number of retrieved patterns in Step 4 is higher than zero, extracts candidate answers from the retrieved relevant passages/documents using the retrieved linguistic patterns. This step was described in Chapter 3 of the present document.

   (b) **Entity Recognition** – Extracts candidate answers from the retrieved relevant passages/documents using the available named entity recognizers.

6. **Answer Extraction – Final Answer Selection** – Selects the final answer.

7. **Answer Presentation** – Shows the answer to the user, accompanied by a supporting passage. Asks the user if s/he believes the answer is correct.[2] If the answer is wrong, the user is prompted to *teach* the system and to provide it the correct answer (for example, it might be the case that the correct answer is stated in the supporting passage, but the system mistakenly extracted/selected a difference piece of information).

8. **Pattern Learning** – If the system has the correct answer to the user question, a new Q/A pair is created with the user question and the answer. New patterns are learned and stored in the knowledge base, associated with the interpreted question. This step is described in Chapter 3 of the present document.

9. Waits for a new question. Go to Step 1.

5.3 Evaluation

5.3.1 Evaluation measures

To evaluate the performance of JUST.ASK, we use the metrics presented in Chapter 2, Section 2.4. However, since learning of patterns is involved and we aim at measuring the performance of the system depending on the number of posed questions, results are calculated as a

[2]Note that in our automatic evaluations of JUST.ASK this step is accomplished by verifying if the answer exists in the reference corpus.

function of the previous questions, which allow us to verify the evolution on the system's performance. Therefore, when evaluating the n^{th} question, we calculate the $recall_n$, $precision_n$, F_1-$measure_n$ and $coverage_n$ as follows:

$$Recall_n = \frac{\#Correctly\ answered\ questions\ till\ question\ n}{n}.$$

$$Precision_n = \frac{\#Correctly\ answered\ questions\ till\ question\ n}{\#Answered\ questions\ till\ question\ n}.$$

$$F_1\text{-}measure_n = \frac{2 \times Precision \times Recall}{Precision + Recall}.$$

$$Coverage_n = \frac{\#Answered\ questions\ till\ question\ n}{n}.$$

5.3.2 Experiments

We test our approach in four scenarios, where we vary the strategies employed to candidate answer extraction:

1. *NER*: JUST.ASK runs with the NER strategies; no (pattern) learning process is involved;

2. *Patterns, with reference as fallback* (P+RF): the pattern-based strategy is used. If a correct answer is found, it is used (with the posed question) as input to a new pattern learning cycle, where new patterns are learned. If no correct answer is found, one answer from the reference corpus is chosen instead;[3] by doing so, we mimic the behavior of a tutor that explicitly provides the system the correct answer.

3. *NER and Patterns* (NER+P): combination of all implemented candidate answer extraction strategies. If a correct answer is found, it is used (with the posed question) as input to a new pattern learning cycle.

4. *NER and Patterns, with reference as fallback* (NER+P+RF): combination of all implemented candidate answer extraction strategies. If a correct answer is found, it is used (with the posed question) as input to a new pattern learning cycle. If no answer is found, an answer from the reference corpus is used.[3]

[3] In this situation, the question is considered as unanswered.

We conducted two different experiments:

Experiment 1 – Iterative learning to answer We assess the system's performance as a function of the number of posed questions in the four scenarios previously described, with $MAX_{pl} = 50$ and $MAX_{ae} = 100$ (recall, from Section 3.4.2, that MAX_{pl} refers to number of passages asked to the search engine from where to learn patterns and MAX_{ae} refers to the number of passages asked to the search engine from where to extract answers);

Experiment 2 – Revising past questions We allow JUST.ASK to revise past wrong or unanswered questions and to solve them using the recently acquired knowledge. For that, we set several milestones in the corpus, at the $(i \times n)^{th}$ questions ($i = \{100, 250, 500\}$, $n \in \{1, ..., 10\}$ and $i * n < 1440$). When JUST.ASK reaches a milestone, it returns to the start and tries to answer all the wrong or unsolved questions till that milestone, using all the learned patterns (except those learned using the question under revision[4]). This process repeats for the subsequent milestones, until the end of the corpus is reached.

As an example, and assuming $i = 250$, when the 250^{th} question is reached, the system goes back to the first question and, using all the learned patterns, it tries to answer all questions until the next milestone (the 500^{th} question), the point where it goes back again to the first question.

Is also worth mentioning that, in our experiments, whenever the pattern-based approach is used to candidate answer extraction (*i.e.*, in scenarios 2, 3 and 4), we start with an empty knowledge base of patterns.

[4]In this experiment, if no correct answer is found by JUST.ASK, an answer from the corpus is used. Therefore, patterns will always be learned for all questions. However, and since we keep track of the question in the origin of each pattern, we do not use the patterns learned from a certain question to find answers to that same question.

5.3.3 Results

5.3.3.1 Experiment 1 – Iterative learning to answer

Table 5.1 presents the performance results of JUST.ASK in all four scenarios. We show the averaged results for the 5 runs and the respective standard deviation σ (when $\sigma > 0.00$). We did not make use of the relaxation strategies. Moreover, recall that JUST.ASK is not selecting answers, since here our goal is to evaluate the system's performance in terms of extracted answers.

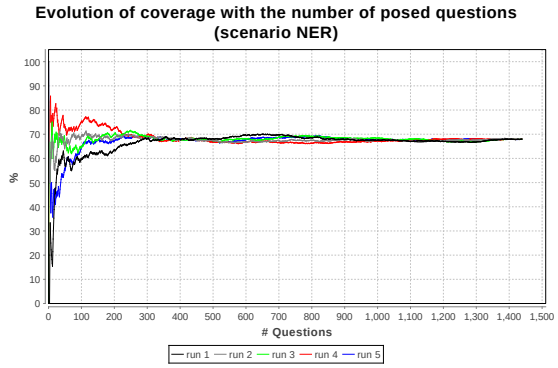
	Scenario			
	1 (NER)	2 (P+RF)	3 (NER+P)	4 (NER+P+RF)
$Recall_{1440}$	55.51%	19.58% ± 0.01	56.58%	57.94%
$Precision_{1440}$	81.69%	61.68% ± 0.02	82.80%	81.21%
F_1-$measure_{1440}$	66.10%	29.73% ± 0.01	67.23%	67.63%
$Coverage_{1440}$	67.96%	31.75% ± 0.01	68.33%	71.35%

Table 5.1: Performance of JUST.ASK when varying the scenario.

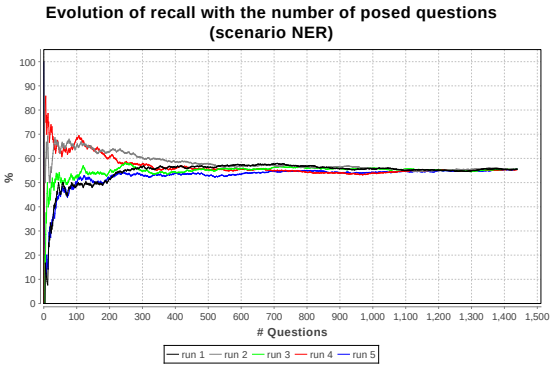
As it can be seen, the success of the system is in large part determined by the use of the NER strategies, which contribute, in average, to nearly 800 correct questions in the total of 1440. JUST.ASK's recall is much lower in scenario 2 (P+RF), when the patterns are used with the reference corpus as fallback, with an average of only 282 questions correctly answered. The use of the patterns in scenario 4 (NER+P+RF) allows an increase of more than 2% in the recall results (around more 35 correct questions) achieved by the NER, when no relaxation strategy is applied.

The graphs in Figure 5.2 allow to make the comparison between scenarios 1 (NER) and 2 (P+RF), by showing the evolution of JUST.ASK's performance with the number of posed questions. In these graphs we compare the results in terms of coverage (number of questions that the system tries to answer) and recall (number of questions correctly answered in the total of questions).

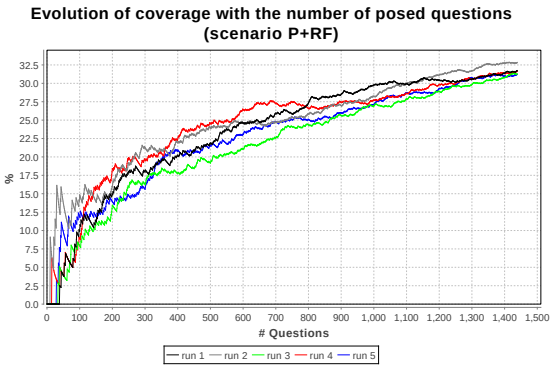
While in the NER scenario both coverage and recall seem to converge to steady values of around 68% and 55% respectively, one can see a clear upward tendency in the scenario



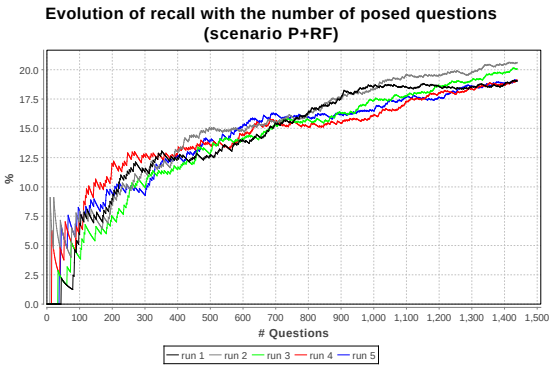
(a) Evolution of coverage, scenario 1 (NER).



(b) Evolution of recall, scenario 1 (NER).



(c) Evolution of coverage, scenario 2 (P+RF).



(d) Evolution of recall, scenario 2 (P+RF).

Figure 5.2: Evolution of JUST.ASK's performance with the number of posed questions, in scenarios 1 (NER) and 2 (P+RF) in the five different runs.

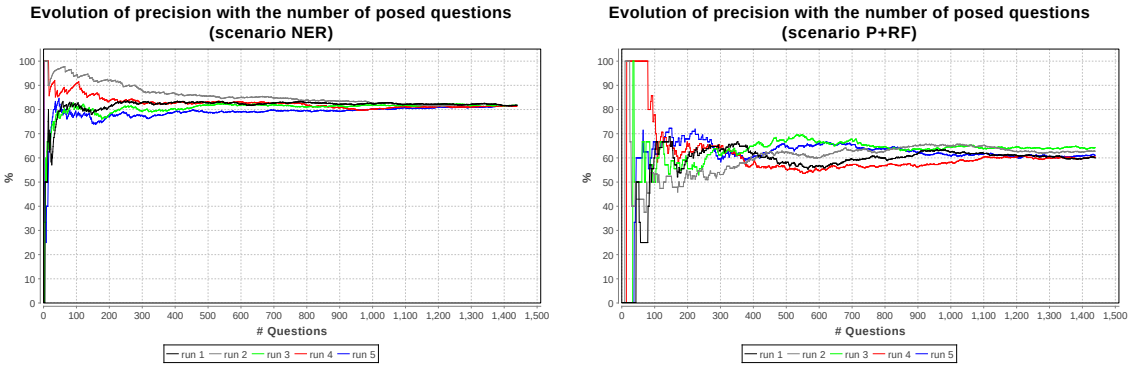
P+RF, regardless of the run, with the number of posed questions. This means that:

1. the system is learning more patterns;
2. the system tries to answer more questions; and,
3. the system is returning more correct answers.

As we can see, this approach is dependent on the order of how the questions appear. For example, while on run 2 the first (correctly) answered question occurs in the 11^{th} interaction (which explains the "bump" in the first interactions in both graphs), on run 1 the first answered question is that of the 45^{th} interaction. Nevertheless, despite of when the system

starts returning correct answers to the questions, results show a consistent increase in recall after some interactions for every run.

Figure 5.3 shows the evolution of the system's precision with the number of posed questions in scenarios 1 (NER) and 2 (P+RF). We can see that, using only NER, the precision is more concentrated near the mean of 80%, while in scenario 2 the precision values are more disperse and tend to converge to a value around 60%.



(a) Evolution of precision, scenario 1 (NER).

(b) Evolution of precision, scenario 2 (P+RF).

Figure 5.3: Evolution of JUST.ASK's precision with the number of posed questions, in scenarios 1 (NER) and 2 (P+RF) in the five different runs.

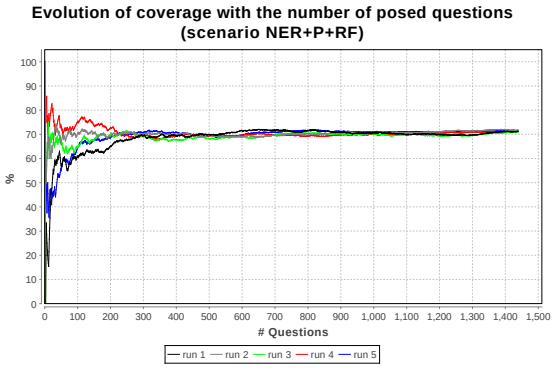
In order to assess the impact of the relaxation strategies in the best scenario (NER+P+RF), we also run JUST.ASK with the relaxation strategies activated. In the end of the test set, at question 1440, the system achieved an average recall of 62.82% ± 0.01 (*vs.* 57.94% without relaxation strategies), precision of 83.27% ± 0.01 (*vs.* 81.21%), F1-measure of 71.61% ± 0.01 (*vs.* 67.63%), and coverage 75.44% (*vs.* 71.35%). Table 5.2 summarizes these results.

	With relaxation	Without relaxation
$Recall_{1440}$	62.82% ± 0.01	57.94%
$Precision_{1440}$	83.27% ± 0.01	81.21%
$F_1\text{-}measure_{1440}$	71.61% ± 0.01	67.63%
$Coverage_{1440}$	75.44%	71.35%

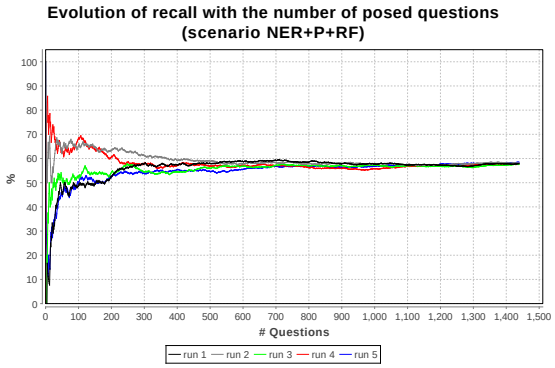
Table 5.2: Impact of the relaxation strategies in the best scenario NER+P+RF.

The graphs in Figure 5.4 allow to compare the results achieved in scenario 4 (NER+P+RF)

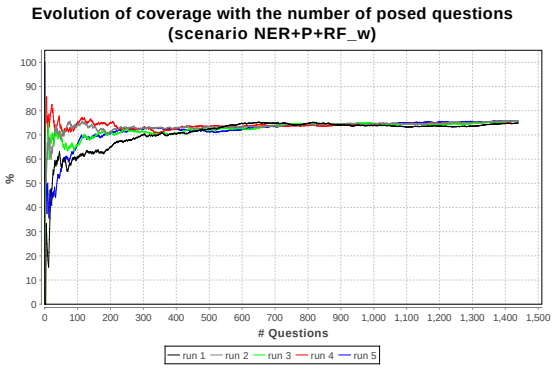
with the number of posed questions, with and without relaxation in the pattern/sentence unification. The main difference between both scenarios lies on the percentage around which the results stabilize after the first 400 questions: recall and coverage are around 7% higher in the scenario where relaxation is used (with an average of more 106 correct questions) than the recall and coverage in the scenario without relaxation. There is also a very slight upward tendency in the coverage when relaxation is applied, since the system tries to give answers to an increasing number of questions.



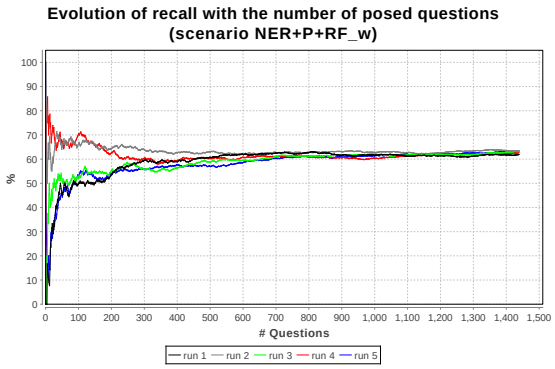
(a) Evolution of coverage, no relaxation.



(b) Evolution of recall, no relaxation.



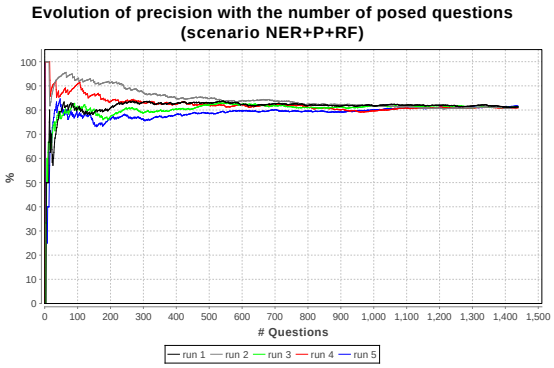
(c) Evolution of coverage, using relaxation.



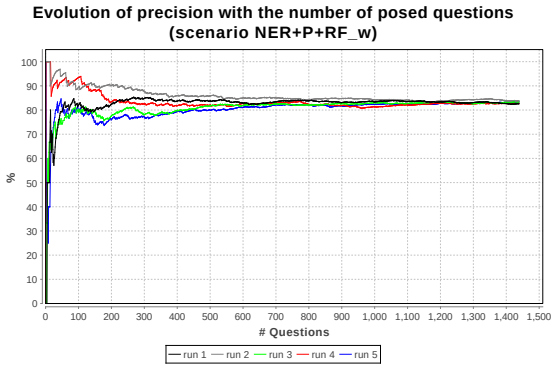
(d) Evolution of recall, using relaxation.

Figure 5.4: Evolution of JUST.ASK's performance with the number of posed questions, in scenario 4 (NER+P+RF) in the five different runs, with and without relaxation.

The evolution of the system's precision in scenario 4 (NER+P+RF), with and without relaxation, is presented in Figure 5.5. Here we can see that there is not too much difference in the precision results with the number of posed questions. Results are slightly higher when using relaxation, however this value is marginal.



(a) Evolution of precision, no relaxation.



(b) Evolution of precision, no relaxation.

Figure 5.5: Evolution of JUST.ASK's precision with the number of posed questions, in scenario 4 (NER+P+RF) in the five different runs, with and without relaxation.

5.3.3.2 Experiment 2 – Revising past questions

Table 5.3 reports the system's average results at the 1440^{th} question when reiterating. It shows the achieved recall, precision, F_1 measure, coverage, and the number of correct answers (between parentheses). Recall that, until the end of the corpus is reached, the system goes back to the first question after $(i \times n)$ questions. Furthermore, the relaxation strategies were not applied in this experiment.[5]

	No Reiteration	**i** 100	250	500
$Recall_{1440}$	19.58% (282)	**23.07%** (332)	22.56% (325)	21.89% (315)
$Precision_{1440}$	**61.68%**	61.36%	60.89%	60.83%
F_1-$measure_{1440}$	29.73%	**33.53%**	32.93%	32.19%
$Coverage_{1440}$	31.75%	**37.60%**	37.11%	35.99%

Table 5.3: Overall performance when reiterating. The system revises past questions at every $(i \times n)$ interactions ($n \in \{1, ..., 10\}$ and $i * n < 1440$).

At it can be seen, JUST.ASK's recall increases when i decreases: the more backward loops, the more correct answers are extracted. When $i = 500$ the system extracts correct answers to 33 questions that it was not able to solve before. When $i = 250$ and $i = 100$, more 43 and 50 questions are solved.

[5]This decision was made due to time constraints since, for instance, when $i = 100$, in the worst case the system has to answer a total 11,940 questions ($= 100 + 200 + ... + 1440$), due to the backward loops.

This situation relates with the fact that our pattern-based approach to candidate answer extraction depends on the order in which the questions are presented to the system: a question q_i that was not correctly answered in the past (because there were no appropriate patterns in the knowledge base, or because the pattern/sentence unification was not successful) might be answerable using the knowledge acquired in a later iteration, after the system has learned patterns from a similar question q_j and its answer (with $j > i$).

Although in this experiment we have tested this idea when only patterns are used for candidate answer extraction, we certainly envision this approach to be applied in the search for the correct answer to a previous question when the NER-based strategies are not successful.

5.3.3.3 Further analysis

Given the achieved results, we further analysed the learned patterns that unified with the sentences. Table 5.4 shows a sample of the patterns, their types, and the questions they tried to solve, as well as the patterns instantiated with the contents of the question (we use ... as placeholder for the answer).

Question 1.	*How many children did Bob Marley have?*		Type of Pattern
Pattern (a)	"NP had CD? NNS"	*Bob Marley had ... children*	WEAK
Pattern (b)	"NP had CD?"	*Bob Marley had ...*	WEAK
Pattern (c)	"NP also VDB NP?"	*Bob Marley also had ...*	INFLECTED
Question 2.	*What does DNA stand for?*		Type of Pattern
Pattern (a)	"NP VBZ for NP?"	*DNA stands for ...*	INFLECTED
Pattern (b)	"NP stand PP NP?"	*DNA stand for ...*	WEAK
Pattern (c)	"NP stands PP NP?"	*DNA stands for ...*	WEAK
Pattern (d)	"NP? - NP"	*... - DNA*	WEAK
Question 3.	*How tall is the Washington Monument?*		Type of Pattern
Pattern (a)	"NP stands NP?"	*the Washington Monument stands ...*	WEAK
Pattern (b)	"NP VBZ about NP?"	*the Washington Monument is about ...*	STRONG
Pattern (c)	"NP? tall, NP"	*... tall, the Washington Monument*	WEAK

Pattern (d)	"NP is NP?"	*the Washington Monument is ...*	WEAK
Question 4.	*How did Eva Peron die?*		Type of Pattern
Pattern (a)	"NP died PP?"	*Eva Peron died ...*	WEAK
Pattern (b)	"NP was VP?"	*Eva Peron was ...*	WEAK
Pattern (c)	"NP VBD of NP?"	*Eva Peron died of ...*	INFLECTED
Pattern (d)	"NP had NP?"	*Eva Peron had ...*	WEAK
Question 5.	*When was Davy Crockett born?*		Type of Pattern
Pattern (a)	"NP_2 VBD_1 VP_3 on NP?"	*Davy Crockett was born on ...*	STRONG
Pattern (b)	"NP_2 was born on NP?"	*Davy Crockett was born on ...*	WEAK
Pattern (c)	"NP_2 VBD_3 in NP?"	*Davy Crockett born in ...*	INFLECTED
Pattern (d)	"NP? , NP_2 VBD_1"	*..., Davy Crockett was*	WEAK
Question 6.	*When was the U.S. capitol built?*		Type of Pattern
Pattern (a)	"NP VBD VP in NP?"	*the U.S. capitol was built in ...*	STRONG
Pattern (b)	"IN? , when NP"	..., when the U.S. capitol	WEAK
Question 7.	*How many sonnets did Shakespeare write?*		Type of Pattern
Pattern (a)	"NP wrote CD?"	*Shakespeare wrote ...*	WEAK
Pattern (b)	"NP VBD over CD?"	*Shakespeare wrote over ...*	
Question 8.	*Where is WWE headquartered?*		Type of Pattern
Pattern (a)	"NP VBD VP in NP?"	*WWE is headquartered in ...*	STRONG
Pattern (b)	"NP, NP?"	*WWE, ...*	WEAK

Table 5.4: Examples of patterns that matched sentences for each given question.

The first thing that is visible is the fact that the pattern context is composed of a few tokens: often only one, but sometimes two (6.(b)) or three (5.(b)).

Many of the patterns are questions rewritings, either at the lexical (5.(b)) or syntactic level (2.(b), 5.(a), 6.(a)), and typically their context is composed of a fairly small number of tokens (less than five). On many patterns the context is composed uniquely by punctuation

marks (2.(d), 5.(d), 8.(b)) or by a function word that does not carry any semantic meaning (3.(b),7.(b))

Some patterns are applicable to questions that expect different answer types (and, therefore, are classified differently), for example, "NP VBD VP in NP?" (5.(c),8.(a)), unified with the sentences retrieved to questions *When was Davy Crockett born?* and *Where is WWE headquartered?*. This kind of patterns can lead to the extraction of answers of a wrong type.

Some patterns provide clues about the semantic type of the extracted candidate answer through the use of adverbs (6.(b),7.(b)) or prepositions (5.(c),8.(a)); in others, the syntax alone allows to fully identify the semantics of the answer (1.(a)); however, most of them extract answers conveying different semantics (4.(d),8.(c)): these are mostly the patterns whose context is composed uniquely by punctuation (2.(d),8.(b)), copular verbs (3.(d)) or the verb *to have* (4.(d)).

The shown patterns capture the linguistic diversity of the different sentence chunks that may contain the answer to the posed question. In this context, we can see that the patterns vary in quality: from those that are too generic and often unify (2.(d),3.(d)) and which can be a source of problems as they typically extract wrong answers (high recall and low precision [Pantel and Pennacchiotti, 2006]) to those that are specific to the posed question and are vey likely to extract the correct answer. These share some characteristics that make them more likely to be successful:

- the pattern context is composed of a relatively small number of words (typically < 5);
- there is a token overlap between the question and the pattern context (5.(b));
- they are of types INFLECTED or STRONG (5.(a));
- the pattern context contains the main verb of the question, or its inflection (2.(b),2.(c),7.(a));
- the pattern or its context contain the headword of the question (1.(a),3.(c))

Overall, patterns allow to discover correct answers to:

Misclassified questions For example, the pattern "NP VBD VP in NP?" extracts the correct answer *the liver* to the question *Where is the bile produced?*, which was misclassified as LOCATION:OTHER instead of ENTITY:BODY. Due to this error in the classification, the NER-based strategies could only extract geography-related answers.

Questions for which there is no appropriate NER strategy For example, the pattern "NP? in NP" extracts the correct answer *American Girl Place* to the question *What is the name of the American Girl store in New York?*, classified as LOCATION:OTHER. Although the question is correctly classified, the used NE-based answer extractor does not recognise *American Girl Place* as a named entity of type LOCATION.

Previous wrong (or unsolved) questions For example, a pattern discovered for a question q_i can be used to extract the answer to a question q_j (with $j < i$). This phenomenon is not possible with the remaining strategies for candidate answer extraction, which are not able to learn from previous interactions.

Moreover, JUST.ASK best results are consistently achieved when the patterns are used in combination with the NER-based strategies for candidate answer extraction.

5.3.4 New results – Just.Ask @TREC-QA_2002-2007

Throughout this thesis, several ideas were tested on JUST.ASK, which allowed the system to evolve and improve. We dedicate this section to an evaluation of the system using all the possible candidate answer extraction strategies, in an optimal way. Given our knowledge about the implemented strategies (acquired from the evaluation results reported in Section 5.3.3.1 and analyzed in Section 5.3.3.3), our approach to candidate answer extraction is described and motivated as follows:

- the NER-based strategies are applied to all questions: JUST.ASK attains a fairly good performance with the NER, which we want to maintain (and improve);

- in the questions for which the NER-based strategies retrieve at least one candidate answer, only the STRONG and INFLECTED patterns are used: the WEAK patterns unify frequently, but are responsible for a large quantity of wrongly extracted candidate answers. As we have mentioned, we wish to maintain JUST.ASK's results achieved with the NER and we do not want to compromise the system's recall by introducing more noise in the answering pipeline (given that the final answer selection in JUST.ASK still requires improvements);

- all patterns types (STRONG, INFLECTED and WEAK) are used in the questions for which the NER did not retrieve any candidate answer: when the NER-based strategies are not effective, we want to explore all the potential of the pattern-based approach to extract candidate answers.

We do not apply relaxation to the patterns, which allows us to directly compare the results with those achieved when the pattern-based approach is not used. Also, since from a previous experiment we noted that a successful pattern/sentence unification depends on the number of tokens in the pattern context (see Section 5.3.3.3), we only used patterns whose context is composed of less than 5 tokens.[6]

The system follows the same pipeline as described on Chapter 2, however here we do not perform clustering in the answers to questions of category NUMERIC, since in a previous experiment we have seen that it was aggregating in the same cluster many incompatible answers (for example, answers that referred to different years).

Table 5.5 presents JUST.ASK's results in the evaluation with the TREC-QA_2002-2007 corpus. Questions were posed to the system in the same order as in run 2 (chosen randomly).

In this table, precision, recall, F_1-measure, MRR and accuracy@1 are measured and shown for the top 3 answers (clusters) returned by the system. For comparison purposes, we show

[6] A set of previous experiments (although of smaller dimension) showed that the use of patterns with more than 5 tokens in the context does not mean an increase of the number of extracted correct answers. Due to time constraints, we decided to use this threshold in the experiments here described, thus discarding all the patterns that do not agree with it.

the results achieved when the pattern-based approach is not used between parentheses. We have manually validated the returned answers.

# Questions		Correct	Wrong	Unanswered
1440		721 (685)	553 (503)	166 (252)
Precision	Recall	F_1-measure	MRR	Accuracy@1
56.6% (57.7%)	50.1% (47.6%)	53.1% (52.1%)	0.43 (0.40)	37.3% (34.7%)

Table 5.5: JUST.ASK best achieved results in the evaluation using the TREC-QA_2002-2007 corpus, after the iterative learning approach has been introduced.

In a total of 1440 questions, JUST.ASK returned answers to nearly 89% (1274 questions). From these, 721 were considered correct and 553 wrong. Therefore, the system's recall was of 50.07%. Moreover, in 540 questions, the correct answer was the one in the highest position of the list of returned answers. As we can see, there is an increment of nearly 5.3% in recall relative to the results when the pattern-based approach to candidate answer extraction is not used, 7.8% in accuracy@1 and 6.5% in MRR. The system returns an answer to many more questions than it did before, but a large percentage of them was considered wrong. This situation results in a drop of precision (compared to when the pattern-based approach is not used) since the number of correct questions does not compensate for the number of questions for which the system gives an answer. Nevertheless, the recall, accuracy@1 and MRR see improvements, as JUST.ASK is able to put more correct answers at the top of the list of returned answers for each question.

In order to better see the contribution of the pattern-approach to candidate answer extraction, in Table 5.6 we show JUST.ASK results according to the different question categories.

Category	Correct	Incorrect	Unanswered	Accuracy
ABBREVIATION	16 (16)	1 (1)	4 (4)	76.19% (76.19%)
DESCRIPTION	20 (0)	35 (0)	45 (100)	20.00% (0.00%)
ENTITY	42 (32)	93 (73)	69 (99)	20.59% (15.69%)
HUMAN	161 (160)	129 (130)	8 (8)	54.03% (53.69%)
LOCATION	203 (203)	90 (90)	0 (0)	69.28% (69.28%)
NUMERIC	279 (274)	205 (209)	40 (41)	53.24% (52.29%)
Total	721 (685)	553 (503)	166 (252)	50.00% (47.50%)

Table 5.6: JUST.ASK results according to the different question categories.

As we can see, the pattern-based approach to candidate answer extraction brought improvements to the accuracy results achieved in almost all question categories. In particular, in category DESCRIPTION, despite the fact that JUST.ASK does not have any strategy dedicated to it. By using the correct answers from the reference, JUST.ASK could learn how to answer this category of questions. A further analysis to the results showed that some of the DESCRIPTION questions were, in fact, misclassified: for example *What part of the Soldiers' anatomy reminded the Indians of the buffalo?* and *What is the lowest point on earth?*, preventing JUST.ASK from extracting candidate answers using the appropriate NER-based strategy. There are, however, two exceptions to the improvements achieved: in categories ABBREVIATION and LOCATION there was no change in the system's performance.

Comparing with the baseline when only the NER-based strategies to candidate answer extraction were used, the analysis of JUST.ASK results allowed us to discover that: a) JUST.ASK does not correctly answer 14 questions that it did before, but b) is now successful in 50 questions (hence the difference of 36 correct questions between the two settings: 684 *vs.* 721). Moreover, in those 14 incorrect questions, in 8 the correct answer was previously at the 3^{rd} (and last) position in the list of returned answers. Finally, the distribution of the now 14 incorrect questions according to the category is the following: LOCATION:4, HUMAN:5, NUMERIC:5 while the distribution of the 50 now correct answers is DESCRIPTION:20, ENTITY:10, HUMAN:6, LOCATION:4, NUMERIC:10.

With the use of the pattern-based approach, JUST.ASK is now extracting more answers, besides those extracted via the NER-based strategies. This means that, although the system is extracting the correct answer from the information sources, in 14 questions the final answer selection is no longer able of successfully choosing it for the top 3 (a result that we have discussed previously: the performance of the final answer selection in JUST.ASK degrades with the number of extracted candidate answers).

A phenomena that could be seen in several questions resulted from the lack of a semantic constraint imposed by the patterns to the extracted answers. For example, JUST.ASK answered *When did Einstein die?* with *the very same date my father was born* and extracted

Israel as candidate answer to *When was the first Kibbutz founded?*. In particular, regarding this last example, we noted that this answer was extracted using a pattern that is applicable to questions with the same syntactic structure, but with different semantic categories. These patterns are often replicated in the system's knowledge base, associated with Q/A seed pairs of different semantics. For example, patterns "NP_2 VBD_3 in $\text{NP}_?$" and "NP_2 VBD_1 VBN_3 in $\text{NP}_?$" can extract a location or a date (or a noun phrase with any other semantics) and are associated with Q/A seed pairs of categories NUMERIC:DATE and LOCATION:CITY (that is, the patterns were learned from seed questions like *Where/when did X die?* and *Where/when was X born?*, respectively).[7]

To deal with this issue, we believe that an extra filtering phase based on the semantics of the extracted answer could be applied when answering questions using these patterns, in order to assure that the answer agrees with the category of the posed question. On the other hand, as we previously remarked in Section 5.3.3.3, the lack of a constraint on the answer's semantics in the learned patterns allowed our approach to cope with, and recover from, (some) classification errors. Therefore, if existing, this filter must to be soft enough not to invalidate the extracted (correct) answers to misclassified questions.

5.4 Discussion

JUST.ASK's new approach to QA is based on a bootstrapping process. The idea behind bootstrapping is to use a set of examples to learn patterns that extract new examples, which in turn are used to learn new patterns. This process continues iteratively, until a stopping condition is reached, and oftentimes a (manual or automatic) validation of the new examples/patterns is performed. Bootstrap is widely used for lexicon and relations induction [Brin, 1999; Pantel and Pennacchiotti, 2006; Jijkoun et al., 2010]. In JUST.ASK, a Q/A seed pair is used to learn (lexico-syntactic) patterns that extract answers which, along with the question, create a new seed that will be used to learn new patterns. This process continues while the system has

[7] Other examples of this type of patterns exist, like the pattern "$\text{NP}_?$ is NP_2", which is associated with categories LOCATION:CITY, DESCRIPTION:DEFINITION, HUMAN:INDIVIDUAL...,

questions to answer. Moreover, and although we do not perform any validation of the learned patterns, if the system returns as final any of extracted answers, they are validated by the user[8] and only the correct ones are used in the next learning cycle.

As mentioned, in JUST.ASK the seeds to the pattern learning algorithm are pairs composed of a question and its respective answer; therefore, in our approach to QA, we assume that the system has a mean to generate correct answers that allows to initiate the bootstrapping process, besides the pattern-based approach. This task can be accomplished through other candidate answer extraction strategies (for example, as in JUST.ASK, based on the extraction of named entities from the relevant passages), used in combination with the pattern-based approach. In the evaluation of JUST.ASK, when testing the system with the NER-based strategies disabled, we resorted to the reference corpus as a provider of the correct answers, which can be viewed as the user explicitly giving the system's the answers, besides asserting the correctness of the automatic responses. As we have seen in the evaluation, by doing so we are allowing the system to learn to answer questions for which it does not have a specific candidate answer extraction strategy, and could not otherwise have answered.

To the best of our knowledge, JUST.ASK is the first QA system that can use answers returned to previous questions in order to (learn patterns to) answer new future questions. JUST.ASK does not require a training phase where patterns are learned to be afterwards applied in extracting candidate answers. Instead, patterns are learned over time with the number of correctly answered questions. In this context, as we saw in the evaluation, the performance of our approach is affected by the sequence in which the questions are posed to the system and by the answers the system chooses as correct. This situation closely relates to the problem of seed initialization in bootstrapping settings [Uszkoreit et al., 2009].

Finally, as we previously mentioned, we envision our pattern-based strategy to Answer Extraction to be applied in combination with others, following a common practice in QA to use multiple and overlapping strategies and resources [Chu-Carroll et al., 2003; Lin, 2007; Ferrucci et al., 2010]. Our approach has the advantage of learning and improving with previously

[8] Or by any automatic mechanism to be integrated in the system.

successful interactions, while not requiring a training corpus and a preceding training phase. Instead, the patterns are iteratively learned and the answering process is improved iteratively.

5.5 Summary

In this chapter we discussed our pattern-based approach to candidate answer extraction in a setting where the QA system is itself the provider of the instances (question/answer) that feed the pattern learning algorithm. After new patterns are learned, they are immediately available to extract new answers. Here we introduced how the data flow in JUST.ASK was changed in order to accommodate the new approach and presented the new system performance. Although most of the categories showed improvements, performance was hurt in questions of category NUMERIC due to the use of ambiguous patterns. These results suggest that a new semantic layer is necessary when applying patterns to extract answers to questions of certain categories.

We envision a system that is able to receive feedback from the user and to use it to improve its internal mechanisms to extract answers to unseen questions. To the best of our knowledge, JUST.ASK is the first system that iteratively learns from successfully answered questions and revises and amends previous questions with newly acquired knowledge.

6 Conclusions and Future Work

Considering the three scientific hypotheses that motivated this work (Chapter 1), three main topics were addressed: candidate answer extraction, final answer selection and iterative QA.

In order to test these hypotheses, the first step was to build a QA system, Just.Ask, to serve as baseline and to be used in our experiments (Chapter 2).

Regarding the candidate answer extraction task (Chapter 3), we proposed an approach based on patterns learned from pairs composed of a question and its correct answer. We described the pattern learning and matching algorithms, and the relaxation strategies applied to cope with the variability and diversity of natural language. We conducted several experiments where we analyzed the impact of this answer extraction approach in Just.Ask.

Concerning the final answer selection (Chapter 4), we presented an approach that takes into account not only the candidate answers' frequency, but also the relations they hold with other candidate answers. Using a limited set of heuristics, encoded mostly in the form of regular expressions, as well as linguistic knowledge from WordNet, we updated the score of every answer. We presented a detailed evaluation and we discussed the impact of frequency, normalization and the semantics relations for the purpose of ranking candidates and selecting the final answer.

Finally, we presented Just.Ask as being able to learn to answer new questions based on previous answers (in Chapter 5), using the described pattern-based approach to candidate answer extraction, in which the QA system is itself the provider of the instances (question/answer) that feed the pattern learning algorithm. After new patterns are learned, they are immediately available to extract new answers. We have presented the new data flow of the system that now allows it to iteratively learn to answer. To the best of our knowledge,

JUST.ASK is the first system that iteratively learns from successfully answered questions and revises and amends previous questions with newly acquired knowledge.

6.1 Contributions

One of the major contributions of this thesis is definitely JUST.ASK, an open-domain QA system, which was fully evaluated and is now freely available for the research community. Besides the system, the test corpora was also made public, which allows JUST.ASK to serve as a baseline against which new QA strategies can be tested and benchmarked.

In order to empirically validate the three scientific hypotheses, a set of ideas were implemented in JUST.ASK. Results supported the validity of the hypotheses.

Besides the contributions stated in Chapter 1, the following list states the published contributions during the course of these PhD studies:

Journal Papers

- Ana Cristina Mendes, Luísa Coheur, João Silva and Hugo Rodrigues. 2013. *Just.Ask – A multi-pronged approach to question answering*, in International Journal on Artificial Intelligence Tools, Vol. 22, n. 1, pp. 1-34
- Ana Cristina Mendes and Luísa Coheur. 2013. *When the Answer comes into Question in Question-Answering: Survey and open issues*, in Journal of Natural Language Engineering, Vol. 19, pp. 1-32
- Sérgio Curto, Ana Cristina Mendes, Luísa Coheur. 2012. *Question Generation based on Lexico-Syntactic Patterns Learned from the Web*, in Dialogue and Discourse, Special Issue on Question Generation, Vol. 2, n. 2, pp. 147-175
- João Silva, Luísa Coheur, Ana Cristina Mendes, Andreas Wichert. 2011. *From symbolic to sub-symbolic information in question classification*, in Artificial Intelligence Review, Vol. 35, pp. 137-154

Papers in International Conferences

- Pedro Fialho, Sérgio Curto, Ana Cristina Mendes and Luísa Coheur. 2012. *Extending a wordnet framework for simplicity and scalability*, in Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)

- Ângela Costa, Tiago Luís, Joana Ribeiro, Ana Cristina Mendes and Luísa Coheur. 2012. *An English-Portuguese parallel corpus of questions: translation guidelines and application in Statistical Machine Translation*, in Proceedings of the Eight International Conference on Language Resources and Evaluation (LREC'12)

- Catarina Moreira, Ana Cristina Mendes, Luísa Coheur and Bruno Martins, 2011, *Towards the rapid development of a natural language understanding module*, in IVA'11 Proceedings of the 10th international conference on Intelligent virtual agents

- Sérgio Curto, Ana Cristina Mendes and Luísa Coheur, 2011, *A minimally supervised approach for question generation: what can we learn from a single seed?*, in EPIA2011 - 15th Portuguese Conference on Artificial Intelligence

- Sérgio Curto, Ana Cristina Mendes and Luísa Coheur, 2011, *Exploring linguistically-rich patterns for question generation*, in UCNLG+EVAL '11 Proceedings of the UCNLG+Eval: Language Generation and Evaluation Workshop

- Ana Cristina Mendes, Luísa Coheur. 2011. *An approach to answer selection in Question Answering based on semantic relations*, in Proceedings of the Twenty-Second international joint conference on Artificial Intelligence (IJCAI)

- Ana Cristina Mendes, Sérgio Curto, Luísa Coheur. 2011. *Bootstrapping Multiple-Choice Tests with The-MENTOR*, in CICLing'11 Proceedings of the 12th international conference on Computational linguistics and intelligent text processing

- Ana Cristina Mendes, Luísa Coheur. 2011. *IT'S ANSWER TIME, Taking the Next in Question-Answering*, in ICAART 2011 - Third International Conference on Agents and Artificial Intelligence

- Ana Cristina Mendes, Luísa Coheur, Paula Vaz Lobo. 2010. *Named Entity Recognition in Questions: Towards a Golden Collection*, in Proceedings of the Seventh International Conference on Language Resources and Evaluation (LREC'10)

- Ana Cristina Mendes, Cláudia Antunes. 2009. *Pattern Mining with Natural Language Processing: An exploratory approach.* in MLDM '09 Proceedings of the 6th International Conference on Machine Learning and Data Mining in Pattern Recognition

- Ana Cristina Mendes, Rui Prada, Luísa Coheur. 2009. *Adapting a Virtual Agent to Users Vocabulary and Needs*, in IVA '09 Proceedings of the 9th International Conference on Intelligent Virtual Agents

All the publications, tools and resources resulting from this thesis are freely available and can be found online at `https://qa.l2f.inesc-id.pt/`.

6.2 Final Discussion

The main advantages and limitations of the approaches followed in this work are summarized in the following (a more detailed discussion can be found in the chapters where each approach was introduced).

Regarding the candidate answer extraction based on patterns learned from question/answer pairs using a minimally supervised approach, we see as advantages the fact that it allows the system to learn to answer based on examples (*i.e.*, question/answer pairs) and that it can be used as an alternative/complement (although not yet sufficiently robust) to traditional approaches to candidate answer extraction based on NER. However, on the downside, this approach relies on the existence of a syntactic analyzer and it tends to over generate (noisy) patterns that lead to the extraction of wrong answers, since some of them are too generic (high recall, low precision). Also, some of the patterns are too specific, which translates in low recall (but high precision) in the answer extraction process.

Concerning the final answer selection based on semantic relations, it has the advantage of

not considering the candidate answers as competitors, leveraging the relations between them to better choose the final answer. As a drawback, it depends on regular expressions and on a semantic network to establish relations between the answers. Moreover, it relies on the assumption that the incorrect answers are not related among themselves, which might not be the case.

Finally, the iterative question answering approach based on previous (correct) answers enables a system to learn with its answers; however, it relies on (user) feedback in order to bootstrap the learning process, which might not be straightforward to get.

Several suggestions to cope with some of the limitations of the developed work are made in the next section, dedicated to future work directions.

6.3 Future Work

To conclude this thesis, several ideas for future work directions are proposed.

Detecting relations between answers Despite the fact that the approach to final answer selection did not bring much improvements to the JUST.ASK (the single-stream setting), the results achieved in the multi-stream QA setting are encouraging. In this topic, other techniques and resources should be explored and tested, for instance, related with paraphrase recognition. Also, it should be important to learn the optimal weights to update an answer's score depending on the relations it holds with others.

Avoiding noisy patterns In the iterative learning to answer approach, the more questions are correctly answered by the system, the more patterns exist to be unified against the relevant sentences. While this increases the chances to extract correct answers to the next questions, it also augments the system's response time. In fact, a fair amount of time used in the pattern-based approach to candidate answer extraction is spent on attempting to unify patterns with the relevant sentences. Therefore, it is necessary to envisage a way to reduce the amount of patterns that will be used in the process of

unification, avoiding to noisy ones. For example, while it is probable that the pattern "NP? [wrote] NP" unifies with a relevant sentence retrieved for the question *Who wrote The Divine Comedy?*, it is very unlikely that the pattern "NP [was painted by] NP?" will do so.

A possible solution would be to resort to machine-learning as a mean to classify each pattern as *unifiable* or not, and using it accordingly. In this context, Section 5.3.3.3 refers to several patterns that successfully unified with (at least) one relevant sentence, and we also identified several characteristics that can motivate the creation of certain features to be used in the classification process.

Promoting useful patterns The existence of too generic patterns, like "NP? [,] NP" are a source of problems, as they often unify with the relevant sentences but typically extract wrong answers (high recall and low precision [Pantel and Pennacchiotti, 2006]). Therefore, it is required to differentiate between the patterns that (often) lead to correct answers to those that introduce noise in the pipeline.

Since the pattern-based approach to candidate answer extraction relies on the user's feedback about the answer, that information should surely be used to score patterns depending on their effectiveness in extracting correct answers. The goal is to award those that lead to correct answers (the good patterns) while penalizing the ones that lead to incorrect answers (the bad patterns). Note that this is a case where negative feedback, stating that the answer was not correct, can be used by the system (as a complement to what was discussed in this thesis, where only the positive feedback was availed by the system for learning new patterns);

Promoting correct candidate answers Another fundamental direction that can be followed has to do with differentiating between the correct extractions and the wrong ones, bringing the former to the first positions of the list of final answers. For example, since it is probable that a correct answer to the question *What does ACLU stand for?*[1]

[1]The expansion of the acronym ACLU, and therefore one correct answer to the question, is *American Civil Liberties Union*.

is composed of capitalised words, it is necessary to better rank these, rather than the ones composed only of digits.

Again, a possible solution could be to resort to classification or regression and identify the probability of each answer being *correct* or not. Among others, the used features could be related with: the answer itself; the strategy that led to the extraction of the candidate answer; the relations it holds with other candidates.

Taking better advantage of the user feedback In this thesis we have maintained a double perspective on the user of a QA system:

- on one hand, s/he is the one that has the information need, that poses questions to fulfil that need, and to which the system has to answer the most accurately possible;
- on the other hand, s/he is also the best possible evaluator of the system's response, provided that s/he is shown a snippet that supports that response.

In the iterative approach to answer questions, we used the user's feedback to build patterns to extract new answers. However, the application of the user's feedback can be taken even further in JUST.ASK:

- it can be used to generate a confidence model of the system's answers;
- it can be used to generate a user's model, in order to allow the system to respond with personalised answers (topic in which the first steps were given by [Quarteroni and Manandhar, 2009]).

Reviewing the system's beliefs Section 5.3.3.2 refers to an experiment in which the system was able to revise previous wrong or unanswered questions. A future work direction could be to use the new answered questions and the newly acquired knowledge (in terms of the patterns learned) to review or consolidate the system's previous beliefs.

There is certainly much work to be done and many research lines to be explored in QA. In this thesis, several efforts were dedicated to the improvement of the answering task. We

hope to have contributed to a new approach in the field that integrates the user's feedback to learn to answer future questions.

Bibliography

Carlos Amaral, Adán Cassan, Helena Figueira, André Martins, Afonso Mendes, Pedro Mendes, Cláudia Pinto, and Daniel Vidal. Priberam's Question Answering System in QA@CLEF 2007. In *Advances in Multilingual and Multimodal Information Retrieval: 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007*, pages 364–371, Berlin, Heidelberg, 2008. Springer-Verlag.

Ion Androutsopoulos, Graeme D. Ritchie, and Peter Thanisch. Natural Language Interfaces to Databases - An Introduction. *Natural Language Engineering*, 1:29–81, 1995.

Ron Artstein. Error return plots. In *12th SIGdial Workshop on Discourse and Dialogue*, Portland, OR, June 2011.

Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. DBpedia: A Nucleus for a Web of Open Data. In *The Semantic Web, Proc. 6^{th} International Semantic Web Conference, 2^{nd} Asian Semantic Web Conference*, ISWC'07/ASWC'07, pages 11–15. Springer, 2007.

Ricardo A. Baeza-Yates and Berthier A. Ribeiro-Neto. *Modern Information Retrieval - the concepts and technology behind search, Second edition.* Pearson Education Ltd., Harlow, England, 2011.

Marco Baroni, Francis Chantree, Adam Kilgarriff, and Serge Sharoff. Cleaneval: a competition for cleaning web pages. In *Proc. 6^{th} International Conference on Language Resources and Evaluation (LREC'08)*. European Language Resources Association (ELRA), 2008.

Phil Blunsom, Krystle Kocik, and James R. Curran. Question Classification with Log-Linear Models. In *Proc. 29^{th} annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '06, pages 615–616. ACM, 2006.

Gosse Bouma, Ismail Fahmi, and Jori Mur. Relation extraction for open and closed domain question answering. In *Interactive Multi-modal Question-Answering*, Theory and Applications of Natural Language Processing, pages 171–197. Springer Berlin Heidelberg, 2011. ISBN 978-3-642-17525-1.

Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, and Andrew Ng. Data-Intensive Question Answering. In *Proc. 10^{th} Text REtrieval Conference*, pages 393–400, 2001.

Sergey Brin. Extracting patterns and relations from the world wide web. In *Selected papers from the International Workshop on The World Wide Web and Databases*, WebDB '98, pages 172–183. Springer-Verlag, 1999. ISBN 3-540-65890-4.

S. Buchholz and W. Daelemans. Complex answers: a case study using a www question answering system. *Nat. Lang. Eng.*, 7:301–323, 2001. ISSN 1351-3249.

Razvan Bunescu and Yunfeng Huang. Towards a general model of answer typing: Question focus identification. In *Proc. 11th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing 2010)*, pages 231–242, 2010.

Claire Cardie, Vincent Ng, David Pierce, and Chris Buckley. Examining the role of statistical and linguistic knowledge sources in a general-knowledge question-answering system. In *Proc. 6th conference on Applied natural language processing*, ANLC '00, pages 180–187, Stroudsburg, PA, USA, 2000. ACL.

Chih-Chung Chang and Chih-Jen Lin. *LIBSVM: a library for support vector machines*, 2001. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

Jennifer Chu-Carroll, Krzysztof Czuba, John Prager, and Abraham Ittycheriah. In Question Answering, Two Heads Are Better Than One. In *Proc. 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 24–31. ACL, 2003.

Charles Clarke and Egidio Terra. Passage Retrieval vs. Document Retrieval for Factoid Question Answering. In *Proc. 26th annual international ACM SIGIR conference on research and development in information retrieval (SIGIR '03)*, pages 427–428. ACM, 2003.

Michael John Collins. *Head-driven statistical models for natural language parsing.* PhD thesis, University of Pennsylvania, Philadelphia, PA, USA, 1999. Supervisor-Marcus, Mitchell P.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. Exploring linguistically-rich patterns for question generation. In *UCNLG+Eval2011, EMNLP 2011 Workshop: Language Generation and Evaluation*, July 2011a.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. A minimally supervised approach for question generation: what can we learn from a single seed? In *15th Portuguese Conference on Artificial Intelligence*, pages 832–844, September 2011b.

Sérgio Curto, Ana Cristina Mendes, and Luísa Coheur. Question Generation based on Lexico-Syntactic Patterns Learned from the Web. *Dialogue & Discourse*, 3(2):147–175, March 2012.

Tiphaine Dalmas and Bonnie Webber. Answer comparison in automated question answering. *Journal of Applied Logic*, 5(1):104–120, 2007.

Hoa Trang Dang, Jimmy J. Lin, and Diane Kelly. Overview of the TREC 2006 Question Answering Track. In *Proc. 15th Text REtrieval Conference (TREC 2006)*, pages 99–116, 2006.

Hoa Trang Dang, Diane Kelly, and Jimmy J. Lin. Overview of the TREC 2007 Question Answering Track. In *Proc. 16th Text REtrieval Conference (TREC 2007)*, 2007.

Yongping Du, Xuanjing Huang, Xin Li, and Lide Wu. A novel pattern learning method for open domain question answering. In *Proc 1st international joint conference on Natural Language Processing*, IJCNLP'04, pages 81–89. Springer-Verlag, 2005.

Christiane Fellbaum, editor. *WordNet: An Electronic Lexical Database.* MIT Press, 1998.

David Ferrucci, Eric Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, William Murdock, Eric Nyberg, John Prager, Nico Schlaefer, and Christopher Welty. Building Watson: An Overview of the DeepQA Project. *AI Magazine*, 31(3): 59–79, 2010.

Jenny Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by Gibbs sampling. In *Proc. 43rd Annual Meeting on Association for Computational Linguistics*, ACL '05, pages 363–370. ACL, 2005.

André Gonçalves. A questão das respostas em sistemas de pergunta/resposta. Master's thesis, Instituto Superior Técnico, Technical University of Lisboa, 2012.

Brigitte Grau, Anne-Laure Ligozat, Isabelle Robba, Anne Vilnat, and Laura Monceaux. FRASQUES: A Question Answering system in the EQueR evaluation campaign. In *Language Resources and Evaluation Conference*, 2006.

Mark A. Greenwood and Robert Gaizauskas. Using a Named Entity Tagger to Generalise Surface Matching Text Patterns for Question Answering. In *Proc. Workshop on Natural Language Processing for Question Answering (EACL03)*, pages 29–34, 2003.

Dan Gusfield. *Algorithms on Strings, Trees, and Sequences - Computer Science and Computational Biology.* Cambridge University Press, 1997.

Sanda Harabagiu, Dan Moldovan, Marius Pasca, Rada Mihalcea, Mihai Surdeanu, Razvan Bunsecu, Roxana Girju, Vasile Rus, and Paul Morarescu. The role of lexico-semantic feedback in open-domain textual question-answering. In *Proc. 39th Annual Meeting of the Association for Computational Linguistics*, pages 282–289. ACL, 2001.

Sanda M. Harabagiu and Steven J. Maiorano. Finding Answers in Large Collections of Texts: Paragraph Indexing + Abductive Inference. In *Proc. AAAI Fall Symposium on Question Answering Systems*, pages 63–71, 1999.

Ulf Hermjakob, Eduard Hovy, and Chin-Yew Lin. Automated question answering in webclopedia: a demonstration. In *Proc. 2nd international conference on Human Language Technology Research*, pages 370–371. Morgan Kaufmann Publishers Inc., 2002.

Andrew Hickl, John Williams, Jeremy Bensley, Kirk Roberts, Ying Shi, and Bryan Rink. Question Answering with LCC's CHAUCER at TREC 2006. In *Proc. 15th Text REtrieval Conference*, TREC 2006. National Institute of Standards and Technology (NIST), 2006.

Lynette Hirschman and Robert Gaizauskas. Natural language question answering: the view from here. *Natural Language Engineering*, 7(4):275–300, 2001.

Zhiheng Huang, Marcus Thint, and Zengchang Qin. Question classification using head words and their hypernyms. In *EMNLP*, pages 927–936, 2008.

Adrian Iftene and Alexandra Balahur. Answer validation on English and Romanian languages. In *Proc. 9th Cross-language evaluation forum conference on Evaluating systems for multilingual and multimodal information access*, CLEF'08, pages 448–451. Springer-Verlag, 2009.

Abraham Ittycheriah, Martin Franz, Whei-Jing Zhu, A. Ratnaparkhi, and R. J. Mammone. IBM's Statistical Question Answering System. In *Proc. 9th Text Retrieval Conference, NIST*, 2001.

Valentin Jijkoun, Maarten de Rijke, and Jori Mur. Information extraction for question answering: improving recall through syntactic patterns. In *Proc. 20th Int. Conf. on Computational Linguistics*, COLING '04. ACL, 2004.

Valentin Jijkoun, Maarten de Rijke, and Wouter Weerkamp. Generating Focused Topic-specific Sentiment Lexicons. In *Proc. 48th Annual Meeting of the Association for Computational Linguistics (ACL 2010)*, pages 585–594, Uppsala, Sweden, 2010. ACL, ACL.

Bert F. Green Jr., Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. Baseball: an automatic question-answerer. In *IRE-AIEE-ACM '61 (Western): Papers presented at the May 9-11, 1961, western joint IRE-AIEE-ACM computer conference*, pages 219–224. ACM, 1961.

John Judge, Aoife Cahill, and Josef van Genabith. Questionbank: creating a corpus of parse-annotated questions. In *Proc. 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 497–504. ACL, 2006.

Daniel Jurafsky and James H. Martin. *Speech and Language Processing (2nd Edition) (Prentice Hall Series in Artificial Intelligence)*. Prentice Hall, 2 edition, 2008.

Boris Katz. Using English for Indexing and Retrieving. Technical report, Massachusetts Institute of Technology, 1988.

Boris Katz. Annotating the World Wide Web Using Natural Language. In *Proc. 5th RIAO Conference on Computer Assisted Information Searching on the Internet*, RIAO '97, 1997.

Mahboob Khalid and Suzan Verberne. Passage retrieval for question answering using sliding windows. In *Coling 2008: Proc. 2nd workshop on Information Retrieval for Question Answering*, IRQA '08, pages 26–33. ACL, 2008.

Jeongwoo Ko, Luo Si, and Eric Nyberg. A Probabilistic Framework for Answer Selection in Question Answering. In *Proc. Human Language Technology Conference of the North American Chapter of the Association of Computational Linguistics, HLT-NAACL*, pages 524–531. ACL, 2007.

Leila Kosseim and Jamileh Yousefi. Improving the performance of question answering with semantically equivalent answer patterns. *Data & Knowledge Engineering*, 66(1):53–67, 2008.

Leila Kosseim, Luc Plamondon, and Louis-Julien Guillemette. Answer formulation for question-answering. In *Proc. 16th Canadian society for computational studies of intelligence conference on Advances in artificial intelligence*, AI'03, pages 24–34. Springer-Verlag, 2003.

Zornitsa Kozareva, Sonia Vázquez, and Andrés Montoyo. Adaptation of a Machine-learning Textual Entailment System to a Multilingual Answer Validation Exercise. In *Working notes of CLEF - ECDL 2006, AVE Workshop*, 2006.

Vijay Krishnan, Sujatha Das, and Soumen Chakrabarti. Enhanced answer type inference from questions using sequential models. In *Proc. conference on Human Language Technology and Empirical Methods in Natural Language Processing*, pages 315–322. ACL, 2005.

Dominique Laurent and Patrick Séguéla Sophie Nègre. QRISTAL, le QR à l'épreuve du public. *Traitement Automatique des Langues*, 46:1–32, 2005.

Vladimir. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

Xin Li and Dan Roth. Learning question classifiers. In *Proc. 19^{th} international conference on Computational linguistics*, pages 1–7. ACL, 2002.

Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2006.

Jimmy Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems*, 25(2), 2007.

Lucian Lita and Jaime Carbonell. Instance-Based Question Answering: A Data-Driven Approach. In *Proc. of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP 2004, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL 2004*, pages 396–403, 2004.

Lucian Lita, Warren Hunt, and Eric Nyberg. Resource analysis for question answering. In *Proc. ACL 2004 on Interactive poster and demonstration sessions*, page 18. ACL, 2004.

Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. Is it the right answer?: exploiting web redundancy for Answer Validation. In *Proc. 40^{th} Annual Meeting on Association for Computational Linguistics*, ACL '02, pages 425–432. ACL, 2002.

Bernardo Magnini, Simone Romagnoli, Alessandro Vallin, Jesús Herrera, Anselmo Peñas, Víctor Peinado, Felisa Verdejo, and Maarten de Rijke. The Multiple Language Question Answering Track at CLEF 2003. In Carol Peters, Julio Gonzalo, Martin Braschler, and Michael Kluck, editors, *CLEF*, volume 3237 of *Lecture Notes in Computer Science*, pages 471–486. Springer, 2003a.

Bernardo Magnini, Simone Romagnoli, Alessandro Vallin, Jesus Herrera, Anselmo Penas, Victor Peinado, Felisa Verdejo, Maarten de Rijke, and Ro Vallin. The multiple language question answering track at clef 2003. In *CLEF 2003. CLEF 2003 Workshop*. Springer-Verlag, 2003b.

Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

Ana Mendes, Luisa Coheur, Nuno Mamede, Ricardo Ribeiro, David Matos, and Fernando Batista. QA@L2F, first steps at QA@CLEF. In *Advances in Multilingual and Multimodal Information Retrieval: 8^{th} Workshop of the Cross-Language Evaluation Forum, CLEF 2007*, volume 5152 of *Lecture Notes in Computer Science*. Springer-Verlag, 2008.

Ana Cristina Mendes and Luísa Coheur. An approach for answer selection in Question Answering based on semantic relations. In *Proc. of the 22nd International Joint Conference of Artificial Intelligence (IJCAI)*, pages 1852–1857. AAAI Press/International Joint Conferences on Artificial Int, 2011.

Ana Cristina Mendes, Luísa Coheu, and Sérgio Curto. From answered questions to Question-Answering. *submitted to Transactions on Speech and Language Processing*, 2013a.

Ana Cristina Mendes, Luísa Coheur, João Silva, and Hugo Rodrigues. Just.ask – a multi-pronged approach to question answering. *International Journal on Artificial Intelligence Tools*, 22(1), 2013b.

Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. The structure and performance of an open-domain question answering system. In *Proc. 38th Annual Meeting on Association for Computational Linguistics*, ACL '00, pages 563–570. ACL, 2000.

Dan Moldovan, Christine Clark, Sanda Harabagiu, and Steve Maiorano. COGEX: a logic prover for question answering. In *Proc. 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology - Volume 1*, NAACL '03, pages 87–93, Stroudsburg, PA, USA, 2003a. ACL.

Dan Moldovan, Marius Paşca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154, 2003b.

Diego Mollá. Learning of graph-based question answering rules. In *Proc. 1st Workshop on Graph Based Methods for Natural Language Processing*, TextGraphs-1, pages 37–44, Stroudsburg, PA, USA, 2006. ACL.

Diego Mollá and Mary Gardiner. Answerfinder: Question Answering by Combining Lexical, Syntactic and Semantic Information. In *Proc. Australasian Language Technology Workshop 2004*, pages 9–16, 2004.

Christof Monz. *From Document Retrieval to Question Answering.* PhD thesis, University of Amsterdam, 2003.

Véronique Moriceau. Answer Generation with Temporal Data Integration. In *Proc. 10th European Workshop Nat. Lang. Generation (ENLG-05)*, pages 197–202, 2005.

Véronique Moriceau. Numerical data integration for cooperative question-answering. In *Proc. Workshop KRAQ'06 on Knowledge and Reasoning for Language Processing*, KRAQ '06, pages 42–49. ACL, 2006.

Véronique Moriceau. *Intégration de données dans un système question-réponse sur le Web.* PhD thesis, Université Paul Sabatier, 2007.

Alessandro Moschitti and Silvia Quarteroni. Linguistic kernels for answer re-ranking in question answering. *Information Processing and Management*, 47(6):825 – 842, 2011.

Alessandro Moschitti, Silvia Quarteroni, Roberto Basili, and Suresh Manandhar. Exploiting Syntactic and Shallow Semantic Kernels for Question Answer Classification. In *Proc. 45th Annual Meeting of the Association of Computational Linguistics*, pages 776–783. ACL, 2007.

Yan Pan, Yong Tang, Luxin Lin, and Yemin Luo. Question classification with semantic tree kernel. In *Pro. 31st annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '08, pages 837–838. ACM, 2008.

Patrick Pantel and Marco Pennacchiotti. Espresso: leveraging generic patterns for automatically harvesting semantic relations. In *Proc. 21st Int. Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 113–120. ACL, 2006.

Anselmo Peñas, Álvaro Rodrigo, Valentín Sama, and Felisa Verdejo. Testing the Reasoning for Question Answering Validation. *Journal of Logic and Computation*, 18(3):459–474, June 2008. ISSN 0955-792X.

Slav Petrov and Dan Klein. Improved Inference for Unlexicalized Parsing. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Proc. of the Main Conference*, pages 404–411. ACL, 2007.

Silvia Quarteroni and Suresh Manandhar. Designing an Interactive Open-domain Question Answering System. *Journal of Natural Language Engineering*, 15(1):73–95, 2009.

Anand Rajaraman and Jeffrey Ullman. *Mining of massive datasets*. Cambridge University Press, Cambridge, 2012.

Deepak Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *ACL '02: Proc. 40th Annual Meeting on Association for Computational Linguistics*, pages 41–47. ACL, 2002.

Dmitri Roussinov, Elena Filatova, Michael Chau, and Jose Robles-Flores. Building on Redundancy: Factoid Question Answering, Robust Retrieval and the "Other". In *Proc. 14th Text REtrieval Conference (TREC 2005)*, 2005.

Luís Sarmento and Eugénio Oliveira. Making RAPOSA (FOX) smarter. In *Working Notes for the CLEF 2007 Workshop*, Berlin, Heidelberg, 2007.

Nico Schlaefer, Petra Gieselmann, and Thomas Schaaf. A pattern learning approach to question answering within the ephyra framework. In *Proc. 9th Int. Conf. on TEXT, SPEECH and DIALOGUE*, 2006.

Stefan Schlobach, David Ahn, Maarten de Rijke, and Valentin Jijkoun. Data-driven Type Checking in Open Domain Question Answering. *Journal of Applied Logic*, 5(1):121–143, 2007.

Dan Shen, Dietrich Klakow, and Geert-Jan Kruijff. Exploring syntactic relation patterns for question answering. In *Proc. 2nd International Joint Conf. on Natural Language Processing*. Springer, 2005.

Hideki Shima and Teruko Mitamura. Bootstrap pattern learning for open-domain clqa. In *Proc. NTCIR-8 Workshop*, 2010.

Anna Shtok, Gideon Dror, Yoelle Maarek, and Idan Szpektor. Learning from the past: answering new questions with past answers. In *Proceedings of the 21st international conference on World Wide Web*, WWW '12, pages 759–768. ACM, 2012.

João Silva. QA+ML@Wikipedia&Google. Master's thesis, Instituto Superior Técnico, Technical University of Lisboa, 2009.

João Silva, Luísa Coheur, Ana Mendes, and Andreas Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 35:137–154, 2011.

Robert Simmons. Answering English questions by computer: a survey. *Communications of the ACM*, 8(1):53–70, 1965.

M. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proc. 10^{th} Text REtrieval Conference (TREC*, pages 293–302, 2001.

Ang Sun, Minghu Jiang, and Yanjun Ma. A maximum entropy model based answer extraction for chinese question answering. In *Proc. 3^{rd} international conference on Fuzzy Systems and Knowledge Discovery*, FSKD'06, pages 1239–1248. Springer-Verlag, 2006.

Stefanie Tellex, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proc. 26^{th} annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '03, pages 41–47. ACM, 2003.

Alberto Téllez-Valero, Antonio Juárez-González, Manuel Montes-y-Gómez, and Luis Villaseñor Pineda. INAOE at QA@CLEF 2008: Evaluating Answer Validation in Spanish Question Answering. In *Working notes of CLEF2008, AVE Workshop*, 2008.

Alberto Téllez-Valero, Manuel Montes-y-Gómez, Luis Villaseñor Pineda, and Anselmo Peñas. Towards Multi-Stream Question Answering using Answer Validation. *Informatica. Special Issue on Computational Linguistics and its Applications*, 34(1), 2010.

Jörg Tiedemann and Jori Mur. Simple is best: experiments with different document segmentation strategies for passage retrieval. In *Coling 2008: Proc. 2^{nd} workshop on Information Retrieval for Question Answering*, IRQA '08, pages 17–25. ACL, 2008.

Hans Uszkoreit, Feiyu Xu, and Hong Li. Analysis and Improvement of Minimally Supervised Machine Learning for Relation Extraction. In *Natural Language Processing and Information Systems, 14^{th} International Conference on Applications of Natural Language to Information Systems, NLDB 2009.* Springer, 2009.

Ellen M. Voorhees. The TREC-8 Question Answering Track Report. In *Proc. 8^{th} Text REtrieval Conference (TREC-8)*, pages 77–82, 1999.

Ellen M. Voorhees. Overview of the TREC 2002 Question Answering Track. In *Proc. 11^{th} Text REtrieval Conference (TREC 2002)*, 2002.

Ellen M. Voorhees. Overview of TREC 2003. In *Proc. 12th Text REtrieval Conference (TREC 2003)*, pages 1–13, 2003a.

Ellen M. Voorhees. Overview of the TREC 2003 Question Answering Track. In *Proc. 12th Text REtrieval Conference (TREC 2003)*, pages 54–68, 2003b.

Ellen M. Voorhees. Overview of the TREC 2004 Question Answering Track. In *Proc. 13th Text REtrieval Conference (TREC 2004)*, 2004.

Ellen M. Voorhees and Hoa Trang Dang. Overview of the TREC 2005 Question Answering Track. In *Proc. 14th Text REtrieval Conference (TREC 2005)*, 2005.

Bonnie Webber, Claire Gardent, and Johan Bos. Position statement: Inference in Question Answering. In *Proc. LREC Workshop on Question Answering: Strategy and Resources*, pages 19–25, 2003.

Bonnie Nash-Webber William Woods, Ron Kaplan. The Lunar Sciences Natural Language Information System: Final Report. Technical report, Bolt Beranek and Newman Inc., 1972.

Fabio Zanzotto and Alessandro Moschitti. Automatic learning of textual entailments with cross-pair similarities. In *Proc. 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, ACL-44, pages 401–408. ACL, 2006.

Dell Zhang and Wee Lee. Question classification using support vector machines. In *Proc. 26th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '03, pages 26–32. ACM, 2003.

Dell Zhang and Wee Sun Lee. Web Based Pattern Mining and Matching Approach to Question Answering. In *Proc. 11th Text REtrieval Conference (TREC 2002)*, 2002.