INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

QA+ML@Wikipedia&Google

João Pedro Carlos Gomes Silva

Dissertação para obtenção do Grau de Mestre em
Engenharia Informática e de Computadores

Júri

Presidente:	Doutora Ana Maria Severino de Almeida e Paiva
Orientador:	Doutora Maria Luísa Torres Ribeiro Marques da Silva Coheur
Co-orientador:	Doutor Andreas Miroslaus Wichert
Vogal:	Doutora Irene Pimenta Rodrigues

Outubro de 2009

Acknowledgements

First and foremost, I would like to express my sincere gratitude to my thesis adviser, Dr. Luísa Coheur, for without her patience, continuous encouragement, and extensive reviewing of this manuscript, much of this thesis would not have come to be. I would also like to thank my engineering confréres for all the wonderful and joyful times we have shared together throughout this academic journey, and for all the wondrous memories that I treasure deeply. Last but not least, a special thanks to my good friend Lala, for never ceasing to care, even with all the kilometers that keep us apart.

To my parents.

Resumo

À medida que a quantidade de informação textual disponível na World Wide Web aumenta, torna-se cada vez mais difícil para o utilizador comum encontrar informação específica de uma maneira conveniente. Por exemplo, encontrar a resposta para uma pergunta tão simples como "*Quem é o homem mais alto do mundo?*", pode-se tornar numa tarefa fastidiosa. Os sistemas de pergunta resposta (*question answering*, em Inglês) oferecem uma solução para este problema, permitindo obter rapidamente respostas sucintas para perguntas colocadas em língua natural. No entanto, a construção destes sistemas exige tipicamente uma quantidade considerável de trabalho repetitivo, moroso, e sujeito a erros humanos, resultando em sistemas caros, e difíceis de adaptar para novos domínios. Para lidar com estes problemas, nesta tese, propomos uma abordagem multi-facetada para *question answering* na Web, com particular foco em técnicas de aprendizagem automática, que permitem ao sistema *aprender* regras, ao invés de ter um perito humano a criá-las manualmente. Em particular, propomos um sistema composto por três componentes: classificação de perguntas, recuperação de passagens, e extracção de respostas. Para o primeiro componente, desenvolvemos um classificador baseado em técnicas de aprendizagem automática, que utiliza um conjunto de *features* lexicais, sintácticas, e semânticas. Para a recuperação de passagens, empregamos uma abordagem multi-estratégia, que selecciona a fonte de informação apropriada, dependendo do tipo da pergunta. Finalmente, para a extracção de respostas, foram utilizadas diversas técnicas de extracção, desde simples expressões regulares, a reconhecedores de entidades mencionadas baseados em técnicas de aprendizagem automática. O sistema foi avaliado com um conjunto de perguntas feitas por potenciais utilizadores do sistema, tendo-se obtido resultados muito prometedores.

Abstract

As the amount of textual information available in the World Wide Web increases, it is becoming harder and harder for regular users to find specific information in a convenient manner. For instance, finding an answer to a simple factual question, such as "*Who is the tallest man in the world?*", can be a fairly tedious task. Web question answering systems offer a solution to this problem by quickly retrieving succinct answers to questions posed in natural language. However, building such systems typically requires a fairly amount of tedious, time-consuming, and error-prone human labor, which leads to systems that are costly, and difficult to adapt to different application domains or languages. To cope with these problems, in this thesis, we propose a multi-pronged approach to web question answering, with a strong focus on machine learning techniques, that allow the system to *learn* rules instead of having a human expert handcrafting them. Particularly, we propose a system comprised of three components: question classification, passage retrieval, and answer extraction. For the first component, we developed a state-of-the-art machine learning-based question classifier, that uses a rich set of lexical, syntactic and semantic features. For passage retrieval, we employ a multi-strategy approach that selects the appropriate information source, depending on the type of the question. Finally, for answer extraction, we utilize several extraction techniques that range from simple regular expressions to automatic machine learning-based named entity recognizers. The system was evaluated using a set of questions that were asked by potential users of the system, yielding very promising results.

Palavras Chave
Keywords

Palavras Chave

Sistemas de pergunta resposta

Aprendizagem automática

Processamento de língua natural

World Wide Web

Keywords

Question answering

Machine learning

Natural language processing

World Wide Web

Contents

1 Introduction **1**

1.1 An introduction to question answering . . . 1

1.1.1 A definition . . . 1

1.2 Challenges . . . 2

1.3 Our approach . . . 3

1.4 Overview of this thesis . . . 4

2 Related work **5**

2.1 Historical overview . . . 5

2.2 The anatomy of a question answering system . . . 7

2.2.1 Question interpretation . . . 7

2.2.1.1 Question classification . . . 8

2.2.1.2 Query formulation . . . 10

2.2.2 Passage retrieval . . . 11

2.2.3 Answer extraction . . . 11

2.3 Machine learning for question answering . . . 12

2.3.1 Naïve Bayesian classifier . . . 14

2.3.2 Support vector machine . . . 16

2.3.3 Sparse network of winnows . . . 23

2.3.4 Bootstrapping . . . 24

2.3.5 Model validation . . . 26

2.4 Evaluation measures . . . 27

2.4.1 Accuracy . . . 27
2.4.2 Precision, Recall, and F-Measure . . . 27
2.4.3 Mean reciprocal rank . . . 28
2.5 Conclusions . . . 29

3 Question classification **31**

3.1 Introduction . . . 31
3.1.1 Our approach . . . 32
3.2 Question type taxonomy . . . 32
3.3 Question feature set . . . 32
3.3.1 Lexical features . . . 32
3.3.1.1 Word level n-grams . . . 34
3.3.1.2 Stemming and Stopword removal . . . 34
3.3.2 Syntactic features . . . 35
3.3.2.1 Question headword . . . 35
3.3.2.2 Part-of-speech tags . . . 40
3.3.3 Semantic features . . . 40
3.3.3.1 Named entities . . . 40
3.3.3.2 Semantic headword . . . 40
3.4 Experimental setup . . . 44
3.4.1 Classification algorithms . . . 44
3.4.2 Data sets . . . 45
3.4.3 Evaluation measures . . . 45
3.5 Experimental results . . . 45
3.5.1 Comparison of classification algorithms . . . 45
3.5.2 Lexico-syntactic features contribution . . . 46
3.5.3 Semantic features contribution . . . 48

3.5.4 Comparison with other works . . . 49
3.5.5 Discussion . . . 51
3.6 Conclusions . . . 53

4 Passage retrieval **55**
4.1 Introduction . . . 55
4.1.1 Our approach . . . 55
4.2 Information sources . . . 56
4.2.1 Google search . . . 56
4.2.2 Wikipedia . . . 56
4.2.3 DBpedia . . . 57
4.3 Query formulation . . . 57
4.3.1 Keyword-based . . . 57
4.3.2 Wikipedia & DBpedia . . . 57
4.3.3 Pattern-based . . . 58
4.4 Query submission . . . 61
4.5 Conclusions . . . 61

5 Answer extraction **63**
5.1 Introduction . . . 63
5.1.1 Our approach . . . 63
5.2 Candidate answer extraction . . . 64
5.2.1 Regular expressions . . . 64
5.2.2 Named entity recognizer . . . 64
5.2.3 WordNet-based recognizer . . . 65
5.2.4 Gazetteer . . . 66
5.3 Answer selection . . . 67
5.3.1 Candidate answer filtering . . . 67

5.3.2 Candidate answer clustering . . . 67
5.3.3 Scoring . . . 68
5.4 Conclusions . . . 68

6 Evaluation **69**
6.1 Experimental setup . . . 69
6.1.1 Data set . . . 69
6.1.2 Evaluation measures . . . 69
6.2 Experimental results . . . 69
6.2.1 Discussion . . . 71
6.2.1.1 Answer ranks . . . 71
6.2.1.2 Unanswered questions . . . 72
6.2.1.3 Practical impact of the proposed techniques . . . 72
6.2.1.4 Web Zeitgeist . . . 73
6.3 Conclusions . . . 73

7 Conclusions and Future work **75**
7.1 Contributions to question answering . . . 75
7.1.1 Question classification . . . 75
7.1.2 Query formulation . . . 75
7.1.3 Answer extraction . . . 76
7.2 Future work . . . 76
7.3 Conclusions . . . 77

I Appendix **79**

A Resources **81**
A.1 WordNet mappings . . . 81

List of Figures

2.1 High-level architecture of a Question Answering system. 7

2.2 Maximum margin hyperplane (MMH) separating positive from negative examples. Support vectors are marked with a circle. 17

2.3 (a) A simple problem in one-dimension that is non-linearly separable. (b) The same problem after mapping into a two-dimensional feature space, using a non-linear mapping $\Phi(\mathbf{x}) = (\mathbf{x}, \mathbf{x}^2)$. In this feature space, the problem is linearly separable. 18

2.4 (a) Tree fragments of the verb phrase (VP) *created the telephone*. (b) Tree fragments of the verb phrase (VP) *built the telephone*. 20

3.1 (a) Parse tree of the question *What is Australia's national flower?*. (b) Parse tree of the question *Name an American made motorcycle.*. The question headword is in bold face. 36

3.2 Examples of parse trees for which headword extraction is non-trivial. (a) Requires the use of a non-trivial rule for *SBARQ*. (b) Requires two non-trivial rules, the first for *SBARQ* and the second for *WHNP*. Trees (c) and (d) both require a post operation fix. 38

3.3 Hypernym tree for the first sense of the synset *actor*. 42

3.4 Examples of questions for which a compound headword is helpful. (a) The compound headword *mountain range* can help to classify the question into LOCATION:MOUNTAIN, while *range* doesn't. (b) The compound headword *capital of Cuba* does not require disambiguation, as opposed to *capital*. 44

3.5 Question classification accuracy using different combinations of lexical and syntactic features, under the coarse-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features. 47

3.6 Question classification accuracy using different combinations of lexical and syntactic features, under the fine-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features. 48

3.7 Question classification accuracy using different combinations of lexical, syntactic, and semantic features, under the coarse-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features. 49

3.8 Question classification accuracy using different combinations of lexical, syntactic, and semantic features, under the fine-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features. 50

4.1 SPARQL query to retrieve the abstract of Wikipedia's *Afonso I of Portugal* article from DBpedia. 58

4.2 (a) Parse tree of the question *Where is the Louvre Museum ?*. (b) Phrasal nodes of the parse tree in (a). 61

5.1 Sample hyponym tree for the synset *animal*. 66

List of Tables

2.1 Li & Roth's two-layer taxonomy for question classification. 9

2.2 Example of semantic classes and respective word list 21

2.3 Number of results for queries of the form *President is a c*, where *c* is a question category. . 22

2.4 Example of patterns and respective precision for the BIRTHDATE question type 26

2.5 Example of the reciprocal ranks for a test set of questions, with correct answers shown in **bold**. 28

3.1 Li&Roth's question type taxonomy, with example questions for each category. 33

3.2 Subset of the head-rules used to determine the question headword, also known as a head percolation table. *Parent* is the non-terminal on the left-hand-side of a production rule. *Direction* specifies whether to search from the left or right end of the rule, either by category first and then by position (by default), or vice-versa when explicitly mentioned. *Priority List* presents the right-hand-side categories ordered by priorities, with the highest priority on the left. 37

3.3 Question patterns to avoid extracting a headword, when it is not needed. 39

3.4 Examples of clusters that aggregate similar synsets together. The complete list can be found in Annex A. 42

3.5 Question classification accuracy using different machine learning algorithms and different training set sizes, under the coarse grained category. 46

3.6 Comparison of question classification results attained by this work, against other results reported in the literature. 51

3.7 Precision and recall for each fine-grained question category, using the feature set of unigrams, question headword, and semantic headword. 52

4.1 Example of extracted question rewrite patterns with respective precision. 61

5.1 Mappings between named entity types and question categories. 65

6.1 List of test questions that were used for the evaluation of the system, plus the rank of the correct answer (zero represents an unanswered question). 70

1 Introduction

We are drowning in a sea of information and yet starving for knowledge.

John Naisbitt, in *Megatrends*

This thesis presents a multi-pronged approach to web question answering, with a strong focus on machine learning techniques and their applicability in the question answering field. The present chapter opens, in Section 1.1, with an introduction to the question answering field, and a description of the problems it attempts to solve. In Section 1.2, a brief discussion introduces the challenges and difficulties inherent to the construction of question answering systems. Section 1.3 presents the approach of this work to web question answering. Finally, the chapter concludes with an overview of this thesis.

1.1 An introduction to question answering

With the advent of the Internet and the World Wide Web (WWW) in the early 1990s, massive amounts of textual information have become widespread available to the general public, making it a highly attractive place for searching information. However, as the amount of information keeps growing at a staggering pace, it is becoming more and more difficult to find specific information. The traditional information retrieval approach to this problem – web search engines –, require a user to pose their *information need* in terms of a set of keywords, which are then used to return a plethora of documents that the engine deems as relevant. While this approach works well in many cases, sometimes, what a user really wants is to find a succinct answer to a given question, instead of searching for it in a collection of thousands of documents. Question answering systems deal with this problem, by providing natural language interfaces in which users can express their information needs in terms of a natural language question, and retrieve the exact answer to that very same question instead of a set of documents.

1.1.1 A definition

Question Answering (QA) is the field of research which aims at building systems that can retrieve succinct, accurate answers to questions posed in natural language, from a (typically large) collection of documents, such as the WWW.

1.2 Challenges

Building a question answering system is not an easy task. QA systems typically require a great deal of human effort, in order to create linguistic rules that can cope with the vast variety of questions that can be asked, and the many different ways in which they can be formulated. For example, consider the question "*What is the capital of Portugal?*", and the different reasonable [1] formulations of the same question:

- Name the capital of Portugal.
- What is Portugal's capital city?
- What's the name of the capital of Portugal?
- Which city is the capital of Portugal?

As can be observed, such a simple question can be formulated in many different ways, making it a very laborious task to manually build specific rules for each different formulation. Moreover, to aggravate the problem, the passages where the answer to any of the above questions may be found can, themselves, be phrased in many different ways:

- Lisbon, the capital of Portugal.
- The capital of Portugal is Lisbon.
- Lisbon became the capital city of Portugal in 1255 due to its central location.

Thus, systems that follow a rule-based strategy tend to target a specific language, resulting in systems that are very difficult to port to different languages. For instance, a previous experiment in question answering carried out in our laboratory (Mendes et al., 2008) – `QA@L2F` –, made use of a great deal of hand-crafted linguistic patterns, which were specific to the Portuguese language, resulting in a system that is difficult to extend and adapt for different languages.

Furthermore, there are also challenges inherent to question answering for the Web. For example, although the Web contains vast amounts of information, it is crucial to provide the right queries to the Web information source, or else we can receive noisy information that will lead the system to extract wrong answers.

[1] Indeed, if we consider ill-formed questions such as *Portugal's capital?*, this problem becomes even more acute.

1.3 Our approach

In order to cope with the aforementioned difficulties, we employ a multi-pronged approach to web question answering, with a strong focus on machine learning techniques that allow the system to learn rules instead of having a human expert handcrafting them. In particular, we propose a system – henceforth dubbed as `QAML` – comprised of three layers, as follows:

Question Classification The question classification layer is responsible for assigning a semantic category to a given question, which represents the type of answer that is being sought after. This task is of prominent importance to an accurate question answering system, as we shall demonstrate in the upcoming chapters. In this thesis, we adopted a machine learning-based approach to question classification, using a rich set of lexical, syntactic, and semantic features. We also evaluated this layer in isolation, so that we could compare our results with others reported in the literature for this task.

Passage Retrieval The goal of the passage retrieval layer is to find relevant passages from the Web, where the answer to a given question might be found. We employed a multi-strategy approach in this layer, using Google search for factual questions, and a combination of Wikipedia and DBpedia for some non-factual questions. Also, we developed a novel, semi-supervised approach, that leverages the redundancy of the Web to learn very precise lexico-syntactic query rewrite patterns, thus augmenting the chance of finding the correct answer and reducing the chance of finding noisy information.

Answer Extraction Once a question has been classified, and relevant passages have been retrieved, the last layer of the system is responsible for extracting candidate answers from the passages, and selecting a final answer. For this layer, we again employed a multi-strategy approach, candidate answer extraction techniques that range from simple regular expressions to automatic machine learning-based named entity recognizers. We also developed a novel approach to extract candidate answers, which makes use of WordNet's hyponymy relations to construct an online exact dictionary matcher, which allows the system to support a wide range of question categories. This layer also makes use of an unsupervised technique to cluster down similar candidate answers together.

Also, our focus will be on factual questions, such as "*Who is the tallest man in the world?*", although we do cover some non-factoid questions, such as ones that ask for a definition – e.g., "*What is an atom?*" – , or a biography – e.g., "*Who was Afonso Henriques?*".

To finalize this section, we also present the main research questions that this thesis attempts to answer:

1. *Can machine learning techniques be applied to the question answering domain, in order to build robust systems that are not tied to a specific language, and do no require much human effort to get good results ?*

2. *How to leverage the vast amount of information available in the Web to accurately answer natural language questions ?*

1.4 Overview of this thesis

This thesis is organized as follows. Chapter 2 surveys related work in the field of question answering. Chapter 3 describes a machine learning-based approach to question classification, as well as its empirical evaluation. Chapter 4 presents the passage retrieval component of the developed system. Chapter 5 describes the last component of the developed system – answer extraction. In Chapter 6, the system's overall performance is evaluated using a test set of questions collected from potential users of the system. Finally, Chapter 7 concludes this thesis by outlining our main contributions to the question answering field, by discussing directions for future work, and by presenting our final remarks.

2 Related work

In this chapter, we provide a literature review on question answering. We begin in Section 2.1 with a brief historical overview of question answering, from its inception in the early 1960s to the present day. Section 2.2 presents a high-level architecture of a modern QA system, as well as a description of their standard components. Section 2.3 covers several machine learning techniques that have been successfully applied to the QA domain. In Section 2.4, we present some of the evaluation metrics that are commonly used to evaluate the performance of a QA system. Finally, in Section 2.5, we draw conclusions.

2.1 Historical overview

Although, in recent years, a great deal of attention has been given to question answering by the research community, QA is by no means a new field of research. In 1965, (R. F. Simmons, 1965) published a survey article describing no less than fifteen question answering systems for the English language, that were built in the preceding five years. Among the reviewed systems, there was BASEBALL (Bert F. Green et al., 1961), which handled questions about baseball games played in the American League over a period of one year. BASEBALL was able to answer questions such as "*Where did the Red Sox play in July 7?*" and "*How many games did the Yankees play in July 7?*". A few years later, (W.A.Woods et al., 1972) – sponsored by NASA – 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 & Gaizauskas, 2001).

Both LUNAR and BASEBALL were essentially natural language interfaces to databases (**NLIDB**), where a user's question is translated into a database query, and the query's output is returned as the answer. These systems were very limited, in the sense that they only worked with a very restricted domain (**closed-domain question answering**), whose knowledge was stored in a database. Additionally, these systems were very hard to port to different domains, and typically required a considerable human effort to build a knowledge base that comprised all the relevant information about the domain. For a detailed overview of NLIDBs up to 1995, refer to (Androutsopoulos, 1995).

Another system that was developed in the same decade was PROTOSYNTHEX (R. Simmons et al.,

1964), which attempted to answer questions from The Golden Book Encyclopedia, a large and syntactically complex natural language text. It was one the first QA systems designed to extract answers from unstructured text, rather than a structured database. The system used a technique that is now commonly referred to as *full-text search*, in which an entry in the index was created for each word in the encyclopedia, except for common words such as *the* and *a*, which were ignored. Moreover, the system also combined words with the same stem such as *cat* and *cats*. The index was then used to retrieve the sentences that most closely resemble the question. For example, given the question "*What do worms eat?*", the sentences "*Birds eat worms on the grass.*" and "*Host worms usually eat grass.*" could be retrieved. Another unique feature of the system was its learning component, in which a human helped to disambiguate some questions or sentences, with the results being stored for further use. PROTOSYNTHEX can be seen as a first step towards a generic question answering system that could work on an unrestricted domain – commonly referred to as an **open-domain question answering** in the QA parlance.

In the following years, many more question answering systems were developed along the lines of LUNAR and BASEBALL, although no significant improvements were made, since the majority of these systems were still limited to restricted-domains.

More recently, with the advent of the WWW in the early 1990s, and the resulting explosion of electronic media, lots of groups began to exploit the Web as a large text corpus, creating the so called **web-based question answering systems**, such as START (Katz, 1988, 1997) [1].

Despite these initial efforts, QA was in some way forgotten for some years, and it wasn't until 1999, with the launch of the QA track (Voorhees, 1999) in the renowned Text REtrieval Conference (TREC) [2], that QA became a very hot research area in the Natural Language Processing (NLP), Information Retrieval (IR) and Information Extraction (IE) communities. In terms of IR, the goal was to promote a shift-away from document retrieval to very short passage retrieval. Ultimately, these passages could be reduced to a short answer to a given question. The IE community also had some interest in QA due, in part, to the fact that the ending of the DARPA sponsored Message Understanding Conferences (MUCs) (Turmo et al., 2006) coincided with the beginning of the QA track at TREC. Moreover, IE also shares some common tasks with QA, such as named entity recognition. As for the NLP community, the QA track revived the interest that began in the 1960s.

Although Question Answering is far from being a finished research area, some research groups are now trying a new wave of QA: interactive question answering systems, *where a dialogue interface enables follow-up and clarification questions* (Quarteroni & Manandhar, 2009).

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

2.2 The anatomy of a question answering system

The general architecture of a modern Question Answering system has become standardized (Jurafsky & Martin, 2008), and it consists of a pipeline made out of three stages: Question Interpretation, Passage Retrieval and Answer Extraction. Figure 2.1 depicts the QA pipeline. In the following three sections, a detailed description of each of these steps will be provided, as well as some common strategies that have been reported in the literature to tackle the problems faced in each.

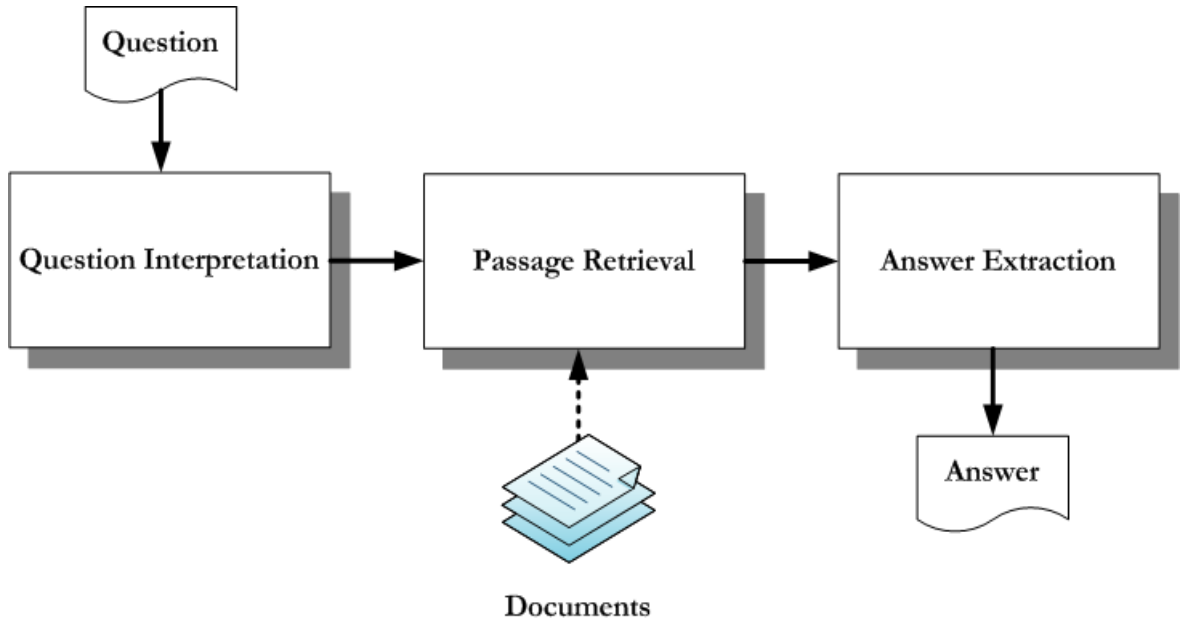

Figure 2.1: High-level architecture of a Question Answering system.

2.2.1 Question interpretation

Question interpretation is the first stage in the QA pipeline. It receives a natural language question as input [3], and tries to extract all the relevant information from it. In order to do so, this stage can be further divided into two sub-tasks: (1) Question Classification, wherein a question is classified into a predefined semantic category (e.g. LOCATION, DESCRIPTION, NUMERIC), that represents the type of information that is expected to be present in the final answer; (2) Query Formulation, in which a question is transformed into a representation that can be used by the Passage Retrieval module to retrieve passages that can contain possible answers[4].

[3] It should be noted that, in case the system supports some kind of dialogue, in which anaphora and ellipsis resolution are needed, this module also needs to receive as input the previous on-going dialogue context.

[4] In the context of this thesis, the focus will be in the transformation of the question into a keyword query that can be feeded as input to an Information Retrieval system, such as Google's search engine.

This stage is of paramount importance to the overall performance of the system, since all the other stages depend upon it. Moreover, an error in this stage, such as a misclassified question, can jeopardize the effectiveness of the overall system. In fact, (Moldovan et al., 2003) showed that about 36.4% of the errors in a QA system are caused by this module.

2.2.1.1 Question classification

As already mentioned, the task of the Question Classification module is to assign a semantic category [5] to a given question. For example, given the questions "*What is the capital of Portugal?*" and "*When was Clint Eastwood born?*", one would expect a question type of `City` and `Date`, respectively. The question type can then be used in the answer extraction stage (described in Section 2.2.3) to narrow down the number of possible candidate answers. Consider again the question "*What is the capital of Portugal?*"; if the system correctly classified it as having the answer type `City`, this would reduce the search space of potential answers, since it would allow us to focus only on those documents that contain cities. Furthermore, for this particular question type, a gazetteer [6] could then be used to make sure only cities were returned as possible answers, a strategy that has proven to be quite effective (Lita et al., 2004).

Formally, a question classifier can be defined as a function f that maps a question $\mathcal{Q}$ into a set of m predefined question categories $\mathcal{C} = \{c_1, c_2, c_3, \ldots, c_m\}$, that is:

$$f : \mathcal{Q} \rightarrow \mathcal{C} \tag{2.1}$$

This definition will be useful when we introduce the machine learning techniques for classification, in Section 2.3.

By the above definition, we can observe that in order to classify a question, we must first have a question type taxonomy according to which questions should be categorized, and secondly, we must devise a strategy to use for the classifier.

Regarding question type taxonomies, several have been proposed in the literature, some of which are hierarchical, some of which are flat. One of the most widely known taxonomies for question classification is Li & Roth's two-layer taxonomy (X. Li & Roth, 2002), which consists of a set of six coarse-grained categories, which are further divided into fifty fined-grained categories, as shown in Table 2.1. With this taxonomy, a question such as "*Which is the highest mountain in the world?*" can be classified into the coarse-grained category LOCATION, or the fined-grained LOCATION:MOUNTAIN [7]. Li's taxonomy is widely used in the machine learning community, probably due to the fact that the authors

[5] Also called question type or question category. These terms will be used interchangeably throughout this thesis.
[6] A gazetteer is a geographical dictionary, tipically used to identify places.
[7] Throughout this thesis, we use this notation when refering to a fine-grained category.

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

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

have published a set of 5500 labeled questions that is freely available on the Web [8], making it a very valuable resource for training machine learning models. Another popular taxonomy is the one used in `Webclopedia` (Hermjakob et al., 2002), a web-based question answering system. `Webclopedia` uses a hierarchical taxonomy spanning over one hundred and eighty categories. To date, it is probably the broadest taxonomy proposed for question classification.

As for the question classifier approaches, they can be classified into two broad categories: rule-based and machine learning approaches. In the former, a set of hand-crafted rules is created by a domain expert, while in the latter, these manual rules are replaced by a typically large set of labeled questions, which is then used to train a model that can predict the question category.

Most of the early approaches to question classification followed the rule-based strategy (Mendes et al., 2008), and used manually built classification rules to determine the question type. `MULDER` (Kwok et al., 2001), for instance, determined the question type just by looking to the question's interrogative pronoun. For example, if the question started with *Who*, it would be classified as HUMAN; if started with *Where*, it would be classified as LOCATION. For systems that deal with very little categories (such as `MULDER`), this approach is feasible and gives very precise results, albeit with very little recall, since the patterns cover only a small set of questions. Other hand-based approaches use a set of complex regular expressions in an attempt to cover the many different variations that a question may have. This approach has, however, many disadvantages. First, it requires a tremendous amount of tedious work in order to create these manual rules. Second, if the system is to be ported to another application domain or even a different language, we would have to manually create a new set of rules, which is not an easy task. Additionally, over time, the rules become very error-prone, and a slight modification of a single rule can cause misclassifications in questions that were, until then, correctly classified.

To overcome some of these difficulties, supervised machine learning approaches to question classification have been devised over the last few years (X. Li & Roth, 2002; Zhang & Lee, 2003; Blunsom et al., 2006). Machine learning approaches tend to achieve a quite high accuracy for the question classification

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

task, with very little effort. Even simple approaches using just surface text features like bag-of-words give remarkable results (Zhang & Lee, 2003). Furthermore, machine learning techniques also two have advantages against the rule-based approach: First, they do not requir expert labor in order to create the patterns; second, they can be easily ported into different application domains, or even different languages. In Section 2.3, some of these techniques will be described to a greater extent.

There are also some questions which are harder to classify than others. Factoid questions [9], for instance, tend to be easy to classify, while questions that ask for descriptions or ones that begin with *What* are much harder (F. Li et al., 2008). Some questions can also be ambiguous, in the sense that they can belong to different semantic classes. For instance, the question "*What is TAP?*" can be classified as ABBREVIATION ("Transportes Aéreos de Portugal") or DESCRIPTION ("*TAP is Portugal's leading airline*").

2.2.1.2 Query formulation

The goal of the query formulation module is to convert a given question to a set of keywords [10], suitable for sending to an information retrieval system, such as Google Search [11] or Yahoo! Search [12]. A naïve approach to query formulation is to send all the keywords that make part of the question – i.e., the whole question – to an IR system. This approach, however, is not very effective, as shown by (Hermjakob et al., 2002). There are several reasons for this to happen. First, IR systems have not been tailored to process natural language questions; second, most IR systems strip some keywords – so called *stopwords* [13] – from the query, effectively eliminating the user's intent. For example, consider the questions "*Who did John Wilkes Booth kill?*" and "*Who killed John Wilkes Booth?*", which have two distinct senses. The first, asks for the person that was killed by John Wilkes Booth, while the second seeks for the person who took John Wilkes Booth's life. When feeded as input to an IR system, these questions will be filtered, and some of its words will most likely be removed (*Who* and *did*), while others will be stemmed (*killed* to *kill*). The final queries for both questions will most likely be the same, completely removing the user's intention.

Another approach that is commonly used is query reformulation, a technique that is particularly effective for web-based question answering systems. This strategy is based on the fact that the Web has lots and lots of redundancy, and the answer to a given question will most likely exist on the web as a reformulation (rephrasing) of the original question, as cleverly exploited in `AskMSR` by (Brill et al., 2002). Consider for example, the question "*What is the capital of Lybia?*". Most likely, in the web, there

[9] Factoid questions ask for a simple fact (hence the name), such as the capital of some country, the name of a person, et cetera.

[10] Again, it is important to clarify that this definition is used in the context of this thesis, and may not hold for other systems. For instance, many offline systems that work by pre-processing a large collection of documents, and storing the information in databases, use the query formulation module to translate a given question into a suitable SQL query that can be used to retrieve the question's answer.

[11] `http://www.google.com`

[12] `http://search.yahoo.com`

[13] A stop word is a typically small, frequently occurring word, with no content or semantic value. The term is a neologism and is believed to have been coined by the computer scientist Hans Peter Luhn, who was an early pioneer in the field of information retrieval.

exists some document that has something along the lines of "*The capital of Lybia is Tripoli*" or '*Tripoli, the capital of Lybia*", making it possible to leverage this redundancy to submit wildcard queries such as "*The capital of Lybia is **" to an IR system. After that, it becomes easier to obtain the answer in the answer extraction stage. Some of these patterns can also be learnt, as will be shown in Section 2.3.4.

2.2.2 Passage retrieval

The goal of the passage retrieval module is to find relevant documents [14] from a given collection. A document is deemed as relevant if it can potentially contain the correct answer for a given question.

The passage retrieval module is sometimes neglected, in the sense that most modern systems use already existing Information Retrieval systems such as Google's search engine to do passage retrieval and so, the real effort is more on the query formulation task than in the passage retrieval one. However, there are still some systems such as the one from `PRIBERAM` (Amaral et al., 2008) and `QA@L2F` (Mendes et al., 2008) that spend a lot of effort in the pre-processing stage, by collecting and organizing information contained in thousands of documents in relational databases.

Besides the question transformation that is carried out by the query formulation module, this module can also make use of the question category that is obtained by the question classification component, by using this information to retrieve only those documents that contain at least one ocurrence of the expected question type, allowing the answer extraction module to only search documents that may potentially have the correct answer.

2.2.3 Answer extraction

The answer extraction module is the last module in the QA pipeline and, therefore, its goal is to extract an answer from the relevant passages returned by the passage retrieval module, and present the answer.

A simple of way of tackling the answer extraction problem is to find frequently occurring patterns in text. Consider, for example, the question "*What is the capital of Portugal?*". In this case, we expect to find snippets of text containing either "*The capital of Portugal is Lisbon.*" or "*Lisbon, the capital of Portugal (...)*". Albeit very simple, these patterns have proven to work very well, as demonstrated in the 10-th edition of QA track of TREC (Voorhees, 2001), where the winning system – (Soubbotin, 2001) – used just a few patterns like these to answer the questions, beating several other systems that used a myriad of deep NLP techniques for answer extraction.

[14] A document can be a web page, a document passage or a structured text. In the context of this thesis, a document will be regarded as a text that can have an answer.

There is, however, a problem with this approach, since it still needs a lot of human effort to hand code all the different rules and variants that a question can have. Consider again the question "*What is the capital of Portugal?*". As already shown in Section 3.1, this very simple question can have so many variations, and it becomes a very difficult task to create and maintain patterns for it.

To deal with this problem, machine learning techniques have been proposed to automatically learn these patterns, with very little or no human effort. In (Ravichandran & Hovy, 2001), the authors propose a method to learn these patterns using bootstrapping, that is, given a known question/answer pair, the authors try to learn answer patterns. Additionally, they propose a simple technique to validate the learnt patterns, so as to make sure only high accuracy patterns are stored. This technique will be described in Section 2.3.4. Although results are very encouraging, especially in the context of web-based question answering, in the TREC QA track however, the learnt patterns do not work so well. The main reason for this is due to the fact that these documents normally do not have the same redundancy as in the Web, and many patterns have long dependencies (many words in between the question keywords and the correct answer) in between, making it a very hard task to construct patterns.

2.3 Machine learning for question answering

Machine Learning is the field of study that is concerned with the question of *how to construct computer programs that automatically improve with experience* (Mitchell, 1997). Within this view, a computer program is said to **learn** from an experience with respect to some task, if the program's performance at the task improves with the experience.

In the 1950s, Arthur Samuel – a pioneer in the field of machine learning –, wrote a checkers-playing program (Samuel, 1959) that was able to learn how to play checkers, by recognizing game patterns that led to wins and game patterns that led to losses. The program learnt to recognize these patterns through experience acquired by repeatedly playing games against a copy of itself, and analyzing the outcome of each move that was played. Samuel's program is now regarded as a milestone in the field of machine learning, and was probably the first computer program that actually learned from experience.

As another example of a learning problem, consider a computer program that deals with question classification, as defined in Section 2.2.1.1. In this setting, the task is to classify questions posed in natural language, the training experience is a data set of questions along with their correct *labels* or *categories*, and a performance measure for this task is **accuracy**, i.e., the percentage of correctly classified questions. In this case, we can say that the program is learning how to classify questions, if it improves its accuracy with the training experience.

Traditionally, the field of machine learning has been divided into three broad categories or learning paradigms: supervised, unsupervised, and reinforcement learning. These are defined as follows:

Supervised learning Supervised learning involves learning a function from a training set of pairs of inputs and corresponding outputs. The learning is said to be supervised because a supervisor is required to direct the learning process, by supplying the desired outputs to the corresponding inputs. More formally, the goal of supervised learning is to learn a function $f\colon \mathcal{X} \rightarrow \mathcal{Y}$, from a training set $\mathcal{D} = \{(x_i, y_i)|x_i \in \mathcal{X}, y_i \in \mathcal{Y}\}_{i=1}^{n}$, in such a way that $f(x) \in \mathcal{Y}$ is a good predictor of the actual value $y \in \mathcal{Y}$. Additionally, when the output of the learnt function is a continuous value, the learning problem is called a **regression** problem; whereas if the output belongs to a discrete set of values, it is called a **classification** problem. As an example, a supervised learning algorithm can be used to tackle the problem of question classification, by learning a function that maps questions into a discrete set of question categories, given a training set of correctly labeled questions.

Unsupervised learning In the unsupervised learning paradigm, the goal is to learn patterns and interesting structures directly from the input, without knowing the corresponding outputs. Unlike supervised learning – which can be seen as a form of *learning by example* –, unsupervised learning techniques do not rely on any *a priori* knowledge, such as training sets of inputs with the corresponding outputs, and can thus be seen as a form of *learning by observation*. The most common example of unsupervised learning is **clustering**, whose goal is to group similar input instances together, into a set of clusters.

Reinforcement learning In reinforcement learning, an agent learns how to act in a given environment, by means of maximizing a reward function. A reward can be seen as a kind of feedback, which allows the learning agent to know if the actions that it is taking are correct or not. By using these rewards, the agent is able to learn a strategy which maximizes the total rewards. This type of learning is typically used in situations where it is very difficult to supervise the learning process, such as playing chess or robotics. The checkers-playing program developed by Arthur Samuel is an example of reinforcement learning.

There is also another learning paradigm – called **semi-supervised learning** –, that falls in-between supervised and unsupervised learning. In semi-supervised learning, both labeled (albeit very little) and unlabeled data is used for training. In Section 2.3.4, we will address a particular technique of semi-supervised learning, usually referred to as bootstrapping.

In the following sections, we present some machine learning techniques that have been successfully applied to question answering. At the end of each section, we will show how these particular techniques have been applied within the question answering domain. To finalize this chapter, we present techniques that can be used to validate the models or hypothesis created by the machine learning techniques.

2.3.1 Naïve Bayesian classifier

The nave Bayesian classifier is a supervised learning method that has been extensively used in many applications involving classification tasks, such as spam filtering in e-mails (Metsis et al., 2006), where it achieves state-of-the-art results, despite its simplifying *naive* assumptions.

Given a fixed set of m categories or cïlasses $\mathcal{C} = (c_1, \ldots, c_m)$, and an input instance represented by a feature vector $\mathbf{x} = (x_1, \ldots, x_n)$, the naïve Bayesian classifier tries to find the most likely class for that instance. This is accomplished by using a probabilistic model that chooses the class c, out of all possible classes, that maximizes the posterior probability $P(c|\mathbf{x})$, i.e., the probability of c given the feature vector $\mathbf{x}$. In statistics, this is often called a **maximum *a posteriori*** (MAP) estimation, and is given by:

$$\hat{c}_{MAP}(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} P(c|\mathbf{x})\,. \tag{2.2}$$

Since it is very difficult to provide a good estimation of $P(c|\mathbf{x})$, Bayes' theorem (also known as Bayes' rule) can be used to to rewrite equation (5.1) as follows:

$$\hat{c}_{MAP}(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} \frac{P(\mathbf{x}|c)\,P(c)}{P(\mathbf{x})}\,. \tag{2.3}$$

Again, estimating the value of $P(\mathbf{x}|c)$ is computationally intractable, as the required number of calculations grows exponentially with the number of features. However, this is where the ***naive* assumption** of the naive Bayesian classifier comes into play, by considering that the features are conditionally independent of one another given the class c. In other words,

$$P(\mathbf{x}|c) = \prod_{i=1}^{n} P(x_i|c). \tag{2.4}$$

By plugging equation (2.4) into equation (2.3), and dropping the denominator $P(\mathbf{x})$ since it does not depend on c – and is therefore constant –, we obtain the final formulation of the naïve Bayesian classifier (NB):

$$\hat{c}_{NB}(\mathbf{x}) = \arg\max_{c \in \mathcal{C}} P(c) \prod_{i=1}^{n} P(\mathbf{x}|c) \tag{2.5}$$

In order to train the classifier, the probabilities of equation (2.5) need to be estimated. This can be achieved using the method of **maximum likelihood estimation** (MLE), in which the probabilities are estimated from relative frequencies in the training set. Let N be the size of this training set, N_c the number of occurrences of class c in the training set, and $N_{x_i,c}$ the number of times feature x_i is seen with

class c. Then, the class priors and the class conditional probabilities are respectively given by

$$P(c) = \frac{N_c}{N} \quad \text{and} \quad P(x_i|c) = \frac{N_{x_i,c}}{N_c}\,. \tag{2.6}$$

There is, however, a problem when using MLE: **zero probabilities**. Since equation (2.5) involves a product of probabilities, a single zero probability will result in the whole product to be zero. For example, if a test example $\mathbf{x}$ happens to have a feature x_i that never appeared in the training set (i.e., $P(x_i|c) = 0\ \forall c$), then the classifier will give zero probabilities for every class, and is therefore unable to classify the example. To avoid this problem, smoothing techniques can be used, such as *add-one smoothing* which simply assumes that every feature x_i is seen at least once. A detailed overview of smoothing techniques can be found in (Jurafsky & Martin, 2008).

Applications to question answering

In the experiments carried out by **(Zhang & Lee, 2003)**, a naïve Bayesian classifier was used for the task of question classification, trained on the standard data set for question classification of Li & Roth, described in Section 2.2.1.1.

A **bag-of-words** model was used to represent a question as an unordered collection of words. Within this model, a dictionary is created, whose length is equal to the number of distinct words present in the training set. After that, questions can be encoded as binary feature vectors $\mathbf{x}$, in such a way that if the question contains the i-th word of the dictionary, then x_i is set to 1; otherwise, x_i is set to 0. To exemplify this model, consider the questions "*What is the capital of Spain?*" and "*Where was Paganini born?*", as well as the resultant word dictionary:

$$[\textit{what}, \textit{was}, \textit{is}, \textit{where}, \textit{the}, \textit{Paganini}, \textit{born}, \textit{capital}, \textit{of}, \textit{Spain}] \tag{2.7}$$

A binary feature vector representation [15] for the question "*What is the capital of Spain?*", would then be:

$$[1, 0, 1, 0, 1, 0, 0, 1, 1, 1] \tag{2.8}$$

Another kind of feature – **bag-of-ngrams** – was also used in the experiments, in order to try to capture some dependencies between words, such as the order in which they appear. With the bag-

[15] Actually, efficient implementations of the bag-of-words model do not explicitly use the binary vector representation, since it is very expensive in terms of memory. This is due to the fact that word dictionaries tend to be quite large and, consequently, the question feature vectors are very sparse, i.e. they have a lot more zeros than ones. A common alternative is to use sparse codes, where the length of our feature vector is equal to the number of words in the question, and the i-th position of the feature vector contains the index of the dictionary word that corresponds to the i-th question word.

of-ngrams features, the classifier showed a slight improvement of 5.8% in terms of accuracy, over the bag-of-words features.

The experimental results showed that naïve Bayesian classifiers trained on 5,500 examples of labeled questions, perform relatively well on the task of question classification, with an accuracy of 83.2% under the coarse grained category. However, the results also showed that in order to achieve this accuracy, the training sets need to be quite large, since when the classifier was trained with just 1,000 examples, the accuracy dropped to merely 53.8%. Additionally, it is important to keep in mind that these results were obtained by using very superficial text features, which leaves some room for improvement.

2.3.2 Support vector machine

Support Vector Machine (SVM), developed by Vladimir Vapnik and his colleagues in the early 1990s (Boser et al., 1992; Cortes & Vapnik, 1995), is a well-known supervised learning algorithm that has been very successfully applied to many problems concerning classification tasks, such as text classification (Joachims, 1997) and, more recently, question classification (Zhang & Lee, 2003). The basic idea of SVM for a two-class learning problem, is to find a linear decision boundary (also called a separating hyperplane[16]) between two classes, which is as far as possible from the closest training instances within each class - the support vectors. Furthermore, this idea can be extended to non-linear decision boundaries using kernel functions. The following formalizes the SVM algorithm for classification.

Linear SVM

In the learning phase, SVM receives a training set $\mathcal{D} = \{(x_i, y_i)|x_i \in \mathbb{R}^d, y_i \in \{-1, +1\}\}_{i=1}^n$ as input, where the instances x_i for which $y_i = +1$ will be referred to as positive examples, whereas the instances x_i for which $y_i = -1$ will be referred to as negative examples. Assuming that the training set $\mathcal{D}$ is linearly separable, then by definition there must exist a hyperplane that completely separates the positive from the negative examples. This hyperplane can be described by

$$\langle \mathbf{w}, \mathbf{x} \rangle + b = 0 \ , \tag{2.9}$$

where $\mathbf{w} \in \mathbb{R}^d$ is a *weight vector* representing the normal to the hyperplane, and $b \in \mathbb{R}$ is a *bias*, that measures the distance from the hyperplane to the origin. There can be, however, more than one such hyperplane, in which case the SVM selects the one for which the distance between it and the nearest training instances in either side is maximized. This distance is called the **margin**, and such hyperplane

[16] A hyperplane is a generalization of the concept of a plane in Euclidean geometry. For instance, in a one-dimensional space a hyperplane is just a point, whereas in a two-dimensional space it is a line.

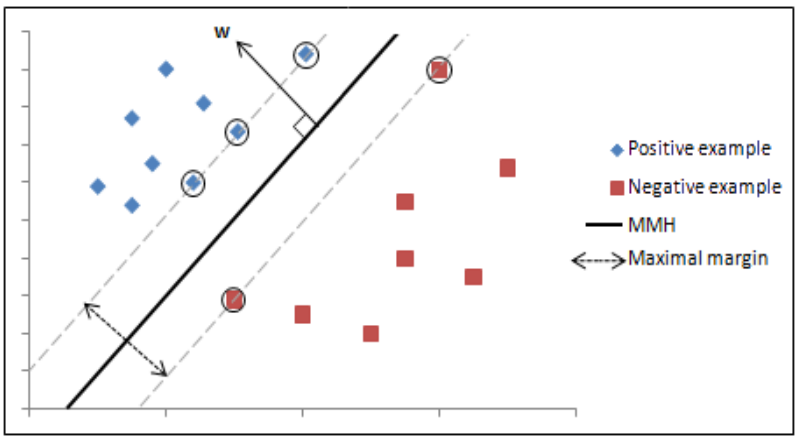

Figure 2.2: Maximum margin hyperplane (MMH) separating positive from negative examples. Support vectors are marked with a circle.

is usually referred to as the **maximum margin hyperplane**. Intuitively, it is expected that the larger the margin, the better the model will generalize to unseen examples.

After finding the maximum margin hyperplane, new input instances are classified based on their relative position to the hyperplane, using the following decision function:

$$f(\mathbf{x}) = sign(\langle \mathbf{w}, \mathbf{x} \rangle + b)\,. \tag{2.10}$$

Thus, every input instance that lies above the separating hyperplane is classified as positive, whereas all input instances that lie below the hyperplane are classified as negative, as depicted in Figure 2.2. Moreover, the following holds for all x_i:

$$\begin{aligned} \langle \mathbf{w}, \mathbf{x}_i \rangle + b \geq 1 \quad \text{for} \quad y_i = +1 \\ \langle \mathbf{w}, \mathbf{x}_i \rangle + b \leq 1 \quad \text{for} \quad y_i = -1\,. \end{aligned} \tag{2.11}$$

The training instances $\mathbf{x}_i$ for which one of the equalities in (2.11) hold - i.e., $\langle \mathbf{w}, \mathbf{x}_i \rangle + b = \pm 1$ -, are the so called **support vectors**. In Figure 2.2, the support vectors are marked with circles. Using geometry, it can be shown that the maximal margin between the two parallel hyperplanes on which the support vectors lie upon is given by $\frac{2}{\|\mathbf{w}\|}$. Therefore, in order to maximize this margin, the goal of SVM is to minimize $\|\mathbf{w}\|$, subject to the constraints in (2.11). This translates into the following quadratic optimization problem:

$$\min_{\mathbf{w},b} \quad \tfrac{1}{2}\|\mathbf{w}\|^2 \tag{2.12}$$

$$\text{subject to} \quad y_i(\langle \mathbf{w}, \mathbf{x}_i \rangle + b) \geq 1 \quad \forall i\,. \tag{2.13}$$

The solution to the above optimization problem yields the maximum margin hyperplane. Moreover, it turns out that this solution depends only on a subset of training instances that lie on the marginal planes - the support vectors. Hence, the decision function of (2.10) can be replaced by the linear SVM formulation:

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{n} \alpha_i y_i \langle \mathbf{x}_i, \mathbf{x} \rangle + b\right), \tag{2.14}$$

where α_i are Lagrange multipliers learnt from the optimization problem, y_i is the label of the support vector $\mathbf{x}_i$, and n is the number of support vectors.

Non-linear SVM

So far, it was assumed that the training set was linearly separable. However, for many real-world problems, this is not usually the case. In order to extend the linear SVM to non-linearly separable problems, the idea is to use a non-linear mapping $\Phi : \mathcal{X} \rightarrow \mathcal{F}$ that can be used to transform the input instances in $\mathcal{X}$ into a higher-dimensional space $\mathcal{F}$ - called **feature space** -, in which the data is linearly separable. Figure 2.3 illustrates one such mapping.

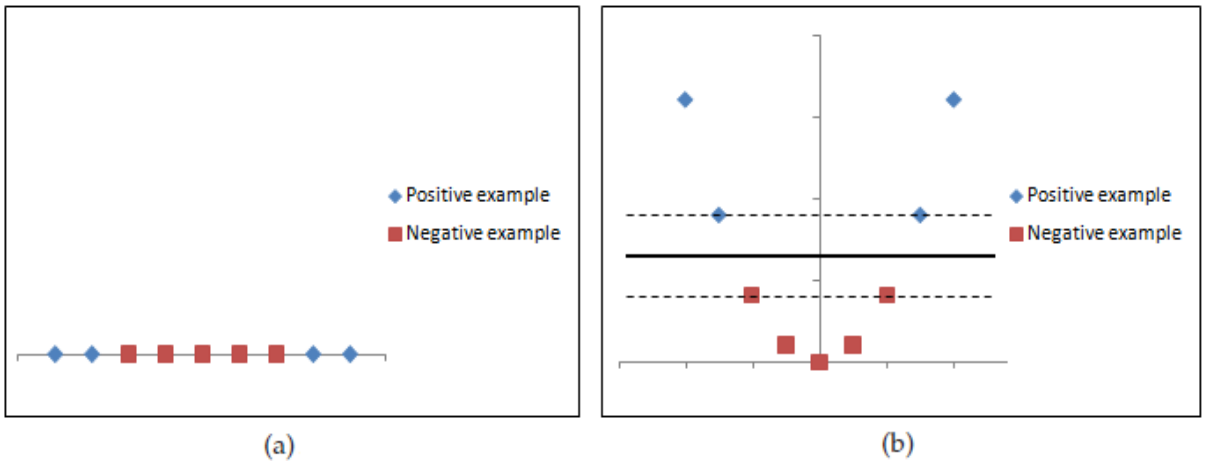

Figure 2.3: (a) A simple problem in one-dimension that is non-linearly separable. (b) The same problem after mapping into a two-dimensional feature space, using a non-linear mapping $\Phi(\mathbf{x}) = (\mathbf{x}, \mathbf{x}^2)$. In this feature space, the problem is linearly separable.

Using the mapping function Φ, the linear formulation of SVM in (2.14) can be modified to deal with non-linear data, by simply replacing the inner product $\langle \mathbf{x}_i, \mathbf{x} \rangle$ with $\langle \Phi(\mathbf{x}_i), \Phi(\mathbf{x}) \rangle$. Furthermore, instead of explicitly doing this mapping, the ***kernel trick*** (Cortes & Vapnik, 1995) can be applied, by replacing the inner product in the feature space with a mathematically equivalent kernel function $K(\cdot, \cdot)$. This yields the non-linear SVM formulation:

$$f(\mathbf{x}) = sign\left(\sum_{i=1}^{n} \alpha_i y_i K(\mathbf{x}_i, \mathbf{x}) + b\right). \tag{2.15}$$

Multi-class SVM

SVM is inherently a binary classifier. However, many real-world classification problems, such as question classification, require classifiers that can classify instances into a set $\mathcal{C}$ containing multiple categories or classes. Thus, SVM needs to be modified in order to deal with multi-class problems. This can be accomplished by combining several binary classifiers (Hsu & Lin, 2001), using one of the following strategies:

- **One-Versus-All (OVA):** In the OVA approach, $|C|$ binary classifiers are built, one for each class, where the i-th classifier is trained with all of the training instances of class c_i labeled with $+1$, and all other training instances of class c_j for $i \neq j$ labeled with -1. Then, for a new input instance, the class c of the classifier that outputs the largest positive value is chosen.
- **All-Versus-All (AVA):** In the AVA approach, $\binom{|C|}{2}$ binary classifiers are trained, one for each possible pair of classes. Classification of new input instances is accomplished by using a *simple majority* voting system, where each pairwise classifier casts one vote to a class c if it classified the input instance as belonging to c, and finally, the most voted class is assigned to the input instance.

Although in (Hsu & Lin, 2001) it is argued that the AVA strategy gives better results, a study by (Rifkin & Klautau, 2004) has shown that the simpler OVA approach can perform as well as any other approach, assuming the binary classifier is well-defined, as is the case of SVM. Nonetheless, both strategies work well in practice.

Applications to question answering

(Zhang & Lee, 2003) compared several machine learning techniques applied to the problem of question classification, with the experimental results showing that, using only surface text features, SVM significantly outperforms every other technique, such as Naive Bayes (Section 2.3.1) and SNoW (Section 2.3.3). However, solely using surface text features may not be enough for some questions. Consider, for example, the following two questions from (Zhang & Lee, 2003):

(2.16) Which university did the president graduate from?

(2.17) Which president is a graduate of the Harvard University?

While both questions clearly have different question categories, a classifier using just bag-of-words features is unable to distinguish the questions, since most words are shared, and the order in which they appear is not taken into consideration. While the bag-of-ngrams features can be used to overcome the problem of word order, it can not be used to weight certain parts of the question that represent the focus

of the question. For instance, the focus of (2.16) is *university* and not *president*, since in the syntactic tree of the question, the depth of *university* is less than the depth of *president*. In order to cope with this problem, the authors proposed a **Tree Kernel**, which measures the similarity between two syntactic trees, by counting the number of common tree fragments. For example, the number of common tree fragments of the syntactic trees [17] depicted in Figure 2.4 is 3.

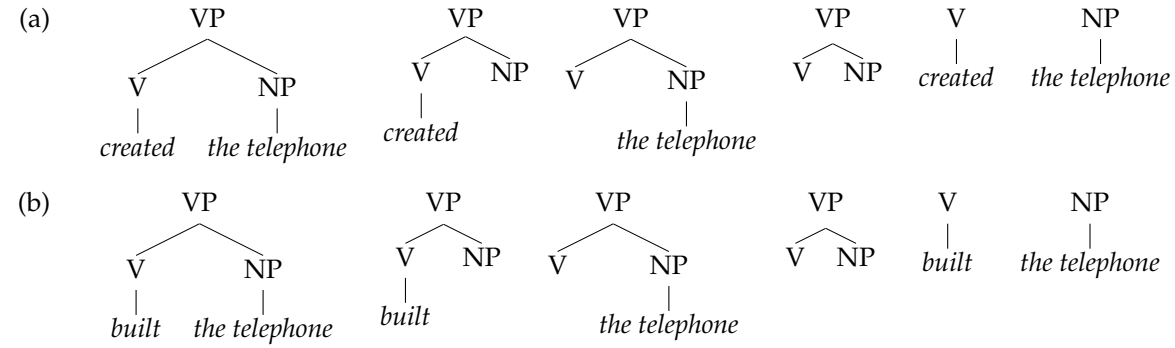

Figure 2.4: (a) Tree fragments of the verb phrase (VP) *created the telephone*. (b) Tree fragments of the verb phrase (VP) *built the telephone*.

Moreover, the authors' kernel also allows tree fragments to be weighted according to their depth in the syntactic tree, so that the question focus receives a larger weight than other fragments that are at the bottom of the tree. The tree kernel is evaluated efficiently using a slightly different version of the dynamic programming algorithm described in (M. Collins & Duffy, 2001). With the tree kernel, the authors achieved 90.0% accuracy for coarse-grained classification.

(Pan et al., 2008) further improved the tree kernel proposed by (Zhang & Lee, 2003), by incorporating semantic features into the kernel. The experiments were carried out in a similar setting, using the same question type taxonomy and training set. The following three types of semantic features were used:

1. **Semantic classes:** A semantic class is a group of semantically similar words, which refer to the same general concept. After analyzing the questions in the training set, the authors manually constructed a set of 20 semantic classes, and gathered a list of words for each class; some of which are shown in Table 2.2. The tree kernel algorithm was then changed, so that two leaf nodes (words) in a parse tree are considered the same, if they belong to the same semantic class.

2. **WordNet** [18] **relations:** WordNet is a lexical database in which words are grouped into sets of synonyms called synsets, that are connected together by means of semantic relations, such as hy-

[17] For simplicity's sake, the noun-phrase (NP) consisting of the article *the* and the noun *telephone* is considered a single node in the tree.

[18] `http://wordnet.princeton.edu/`

Table 2.2: Example of semantic classes and respective word list

Semantic class	Word list
PERSON	man, daughter, actor, resident, robber
CREATURE	animal, plant, flower, tree, tiger
LOCATION	airport, bridge, state, city, park, street
ORGANIZATION	company, bank, club, college, army
SUBSTANCE	stone, mineral, fuel, gas

pernymy [19] and antonymy[20]. The relations of synonymy and hypernymy were integrated directly into the tree kernel, by giving a similarity score to two leaf nodes n_1 and n_2, if n_1 is synonym of n_2, or if n_1 is a direct hypernym of n_2, or vice-versa. For instance, using this type of feature, the number of shared tree fragments of the syntactic trees in Figure 2.4 would be 6 (out of 6), because according to WordNet, *create* is a direct hypernym of *build*.

3. **Named entities:** Only three types of named entities were used: NE_PERSON, NE_ORGANIZATION and NE_LOCATION.

While the first two semantic classes were directly incorporated into the tree kernel, named entities were only used in a pre-processing stage of the questions in the training set. In this stage, every extracted named entity was replaced with the corresponding named entity type. Also, for every noun-phrase in a question, only the headword was kept. For example, given the question "*What is [NP [the only color]] Johnny Cash wears on stage?*", the pre-processing stage transforms it into "*What is* ***color*** **NE_PERSON** *wears on stage?*". The experimental results evidenced that semantic classes just by themselves are very helpful to question classification, resulting in an accuracy of 93.2%. Moreover, semantic classes can also be used as a form of augmenting the training set, as demonstrated by (Bhagat et al., 2005). With all semantic features combined, the authors achieved an accuracy of 94.0% which, to date, outperforms every other question classifier on the standard training set of Li & Roth, for coarse-grained classification.

(Krishnan et al., 2005) and **(Huang et al., 2008)** used the notion of *informer span* and *headword* as features to train a SVM. The authors theorize that a question can be accurately classified using very few words, which correspond to the object of the question. The main difference between the two approaches is that in the former, a sequence of words – the informer span – is considered, while in the latter, only a single word is taken into account – the headword. For example, considering the question "*What is the capital of Italy?*", the informer span is *capital of Italy* and the headword is *capital*. Both works evaluated their experiments using the training set of Li & Roth, with (Krishnan et al., 2005) reaching an accuracy

[19] Hypernymy is a semantic relation between a more generic word - hypernym or superordinate -, and a more specific word - hyponym. For example, a flower is a hypernym of sunflower, while, conversely, sunflower is a hyponym of flower.
[20] Antonymy is a semantic relation between two words that can, in a given context, convey opposite meanings.

Table 2.3: Number of results for queries of the form *President is a c*, where *c* is a question category.

Query	Number of results	Normalized results
President is a person	259	0.8662
President is a place	9	0.0301
President is an organization	11	0.0368
President is a measure	20	0.0669
President is a date	0	0

of 93.4% and 86.2% for coarse- and fine-grained classification, respectively, while (Huang et al., 2008) attained 93.4% and 89.2%.

(Moschitti & Basili, 2006) used tree kernels for both question and answer classification. As for question classification, the experiment was carried out in a similar setting of (Zhang & Lee, 2003), although the authors used a slightly modified question type taxonomy, resulting in significantly worse results. For answer classification, the authors manually annotated a set of paragraphs with the categories DESCRIPTION, DEFINITION and MANNER; which were used to train the SVM. Then, given a new paragraph, the SVM classified it as belonging to one of the three above categories. This information could then be used to validate answer candidates according to the expected question type, i.e., a correct answer for the question type DESCRIPTION should be classified as DESCRIPTION by the answer classifier.

(Solorio et al., 2004) exploited the Web's redundancy to automatically extract question features, which were then used to train a language independent question classifier using SVM. Their method is best explained by means of an example. Considering the question "*Who is the president of the French Republic?*", the authors use a set of heuristics to extract the focus of the question – the word *president* –, and then submit several queries to Google's search engine, one for each possible question category, as shown in Table 2.3. As expected, the query for the question category PERSON is the one which returns more results for this particular question. Finally, the number of returned results for each query are counted and normalized, with the resulting values being subsequently used as features in the learning algorithm. For the Portuguese language (Solorio et al., 2005), the authors achieved a classification accuracy of 78.1%, when using a combination of the Web extracted features and a type of feature called Prefix-4 (prefixes of size 4), which can be seen as a naive, language independent form of stemming [21]. Moreover, the results were obtained using a small training set of just 180 questions collected from CLEF.

[21] Stemming is a technique that reduces words to their grammatical roots or stems, by removing their affixes. For instance, after applying stemming, the words *learned* and *learning* both become *learn*.

2.3.3 Sparse network of winnows

The Sparse Network of Winnows[22] (**SNoW**) learning architecture (Carlson et al., 1999) is a multi-class classifier that is particularly adequate for learning problems that typically involve a large number of features, which may not be known *a priori*.

In SNoW's most basic configuration, a two-layer network is created, with an input layer comprising a set of nodes, each representing a distinct feature in the training set; and an output layer consisting of a set of target nodes, one for each possible class label. This setting resembles a single-layer neural network. The objective is then to find a set of weighted edges connecting the target nodes to the input features, with the restriction that an edge $w_{t,i}$ is allocated and linked, if and only if the i-th feature is seen associated with target t in the training set (hence the name **sparse network**). Conceptually, $w_{t,i}$ represents the importance of the i-th feature for the classification of the class associated with t.

The network is then fed with a set of training examples for which the correct class label is known, and each target node is individually trained, using the OVA multi-class strategy described in Section 2.3.2. In this learning stage, the algorithm passes over the training examples, predicting the output of each, and comparing it to the desired output. For each misclassified example, SNoW updates the weight vector of the corresponding target node, using an **update rule** based on one of the following learning algorithms: Naive Bayes (Section 2.3.1), Perceptron (Mitchell, 1997), and Winnow (Littlestone, 1987) - an algorithm that is closely related to the Perceptron (in the sense that it also learns a linear decision hyperplane), but has a much better performance when many features are irrelevant.

Finally, given a new test example, SNoW calculates the activation value of each target node, by means of the inner product between the node's weight vector and the feature vector of the test example. The algorithm then uses a *winner-takes-all* voting system, in which the target node with the largest activation output becomes the prediction for the given test example.

Applications to question answering

(X. Li & Roth, 2002) developed a hierarchical question classifier based on the SNoW learning architecture, as well as the homonymous two-layer question type taxonomy that is widely employed in question classification. The hierarchical classifier is composed of two simple SNoW classifiers with the Winnow update rule, where the first classifier is used to classify a question into coarse grained categories, and the second into fine grained ones. The SNoW algorithm described in the previous section is also modified in order to allow the classifiers to predict more than one category.

[22] Available at `http://l2r.cs.uiuc.edu/~danr/snow.html`

A feature extractor is used to automatically extract primitive features from each question, ranging from syntactic features such as *words*, *part-of-speech tags* and *chunks* (non-overlapping phrases), to semantic features like *named entities* and *semantically related words* (words that often appear associated with a specific question category). The latest is constructed semi-automatically, and is closely related to the concept of semantic classes utilized by (Bhagat et al., 2005; Pan et al., 2008). For example, for the category LOCATION:MOUNTAIN, the semantically related words are *highest*, *volcano*, *lava*, *peak*, among others. Also, a set of complex features are also composed over the primitive features, such as n-grams, which is defined over the primitive word features.

The training set used for the evaluation of the classifier consisted of 5500 labeled questions[23], and a test set of 500 questions collected from the 10-th edition of TREC. Although the attained results were over-the-top, they can be misleading since the evaluation metric is not accuracy. The authors used a metric called $P_{\leq 5}$, which defines that a question is correctly classified, if its true category is present in the list of five categories returned by the classifier. Using this measure, the reported results achieved $P_{\leq 5} = 98.8\%$ and $P_{\leq 5} = 95.0\%$ for coarse, and fine grained classification, respectively. However, when using *real* accuracy – i.e., $P_{\leq 1}$ –, the results dropped accordingly to 91.0% and 84.2%.

Also, the results showed that the semantically related words is the most prominent feature, and that the use of a hierarchical classifier over a flat classifier provides no significant advantages.

In **(F. Li et al., 2008)**, the authors went on to further improve their former SNoW based question classifier, by including even more semantic features, such as WordNet senses and more named entities. However, the experimental results showed that only a slight accuracy improvement of roughly 1% was attained, over the coarse and fine grained categories.

2.3.4 Bootstrapping

Bootstrapping refers to a particular type of semi-supervised learning algorithms, which starts with a small set of annotated **seed examples**, and then iteratively learns new hypothesis that can be used to generate a potentially large set of new examples (Turmo et al., 2006).

It is particularly useful for natural language processing problems for which the amount of available annotated data is very scarce, and its acquisition is either infeasible, or very costly. In fact, it was its successful usage for the task of word-sense disambiguation (Yarowsky, 1995) – a classic NLP problem, for which a large corpora of annotated data is typically required –, that triggered interest in bootstrapping.

[23]The experimental data is available at `http://l2r.cs.uiuc.edu/~cogcomp/Data/QA/QC/`

Applications to question answering

As already pointed out in Section 2.2.3, a successful strategy to extract exact answers for a given question is to define a set of surface text patterns that represent the different forms in which the answer can appear. For example, for question like "*When was* NAME *born?*", typical answers can be found in the form of the following patterns:

```
<NAME> was born in <ANSWER>
<NAME> (<ANSWER>-.
```

(Ravichandran & Hovy, 2001) developed a method to learn patterns like the above, using bootstrapping. The technique works in two-phases, where the first (Algorithm 1) is used to acquire patterns from a set of seed examples, and the second validates the learnt patterns by calculating their precision.

Algorithm 1 Pattern-learning using Bootstrapping

1: A seed example (Q, A) where Q is the question term and A is the answer term
2: Submit the (Q,A) pair to a search engine
3: Retrieve the top 1000 returned documents
4: Apply a sentence breaker to the documents, and retain only those that contain both the question and the answer term
5: Tokenize the sentences into words, and filter ill-formed content such as HTML tags
6: For each sentence, find all possible substrings, along with their counts
7: Retain only those substrings that contain both the question and the answer term
8: Take the longest matching substring with the highest count, and replace the question and the answer term by the tags ⟨NAME⟩ and ⟨ANSWER⟩, respectively
9: Return the transformed substring

To better illustrate the pattern-learning algorithm, consider a worked example from (Ravichandran & Hovy, 2001), that uses the seed example $(Mozart, 1876)$ for the induction of patterns for question type BIRTHDATE. The following steps are carried out:

1: Steps 2-4 of Algorithm 1 are carried out, resulting in the following three sentences:

   (i) *The great composer* ***Mozart (1756-1791)*** *achieved fame at a young age.*

   (ii) ***Mozart (1756-1791)*** *was a genius.*

   (iii) *The whole world would always be indebted to the great music of* ***Mozart (1756-1791)****.*

2: Since the longest matching substring for all three sentences is "*Mozart (1756-1791)*" (with count 3), the pattern "⟨*NAME*⟩ *(*⟨*ANSWER*⟩ -" is returned[24].

The authors then repeat the above procedure for different seed examples of the same question type, such as "*(Gandhi, 1869)*" and "*(Newton, 1642)*", and calculate the returned patterns precision. The precision is calculated by submitting only the question term to a search engine, and then dividing the number

[24]The string "*1791)*" is assumed to be removed in a post-processing stage, although the authors do not refer this step.

of times the pattern matches with the correct answer term, by the number of times it does not. For example, to calculate the precision of the BIRTHDATE pattern "⟨*NAME*⟩ *was born in* ⟨*ANSWER*⟩", the query "*Mozart*" is submitted to a search engine; then, the precision of the pattern would be $\frac{1}{2}$ if the retrieved sentences were the following:

- *Mozart was born in Salzburg.*
- *Mozart was born in 1756.*

Table 2.4 shows some of the patterns acquired by (Ravichandran & Hovy, 2001), along with their respective precision.

Table 2.4: Example of patterns and respective precision for the BIRTHDATE question type

Precision	**Pattern**
1.0	⟨NAME⟩ (⟨ANSWER⟩ -
0.85	⟨NAME⟩ was born on ⟨ANSWER⟩
0.6	⟨NAME⟩ was born in ⟨ANSWER⟩
0.59	⟨NAME⟩ was born ⟨ANSWER⟩

(Wang et al., 2005) also used a similar bootstrapping technique (although the authors argue that the method is unsupervised) for the task of query formulation. Their goal was to learn patterns that could be used for transforming a question into an effective keyword query.

2.3.5 Model validation

The motivation for model validation techniques is two-folded: (1) Model Selection, where the goal is to estimate the best parameters for a given technique, such as the number of neighbors (k) in k-NN ((Mitchell, 1997)); (2) Performance Estimation, in order to do a reliable estimation of the accuracy of a model. In this section, two of the most widely used validation techniques will be briefly [25] described:

- **Holdout method:** The holdout method works by partitioning the data set $\mathcal{D}$ into two independent sets, usually referred to as **training set** and **test set**. The former, is used by the learning algorithm to derive a model, whereas the latest is used to estimate the accuracy of the created model. It is common practice to allocate two-thirds of the data set to the training set, and one-third to the test set (Kohavi, 1995).
- **Cross-validation:** In k-fold cross-validation, the data set $\mathcal{D}$ is randomly partitioned into k independent subsets or *folds* of approximately equal size. The learning algorithm is then trained and

[25] For an in-depth description of validation techniques, refer to (Kohavi, 1995).

tested k times, where, in the i-th iteration, the subset $\mathcal{D}_i$ is used for testing, and all the remaining subsets $\mathcal{D} \setminus \mathcal{D}_i$ are used for training. When the value of k is equal to the total size of the data set, the method usually goes by the name of **leave-one-out**.

2.4 Evaluation measures

In order to assess the overall performance of a question answering system, several evaluation measures have been proposed in the literature. The following sections will cover the most popular evaluation measures for question answering, that have been extensively used in QA evaluation forums.

2.4.1 Accuracy

Accuracy is defined as the number of correctly answered questions, divided by the total number of questions in a test collection i.e,

$$Accuracy = \frac{\text{\#Correctly answered questions}}{\text{\#Total questions}} . \tag{2.18}$$

Throughout the years, accuracy has been used as the main measure for the evaluation of systems participating in the QA track of both CLEF and TREC (Vallin et al., 2006; Forner et al., 2008; Dang et al., 2008), although sometimes with slight variations with respect to the above definition.

2.4.2 Precision, Recall, and F-Measure

The two most widely used evaluation measures from information retrieval (Baeza-Yates & Ribeiro-Neto, 1999) - precision and recall -, have also been considered for some question answering tasks, that typically require more than one correct answer for a given question. For instance, in the QA track of TREC (Dang et al., 2008), these metrics were used to evaluate system performance for LIST type questions, such as "*What are the names of the Grimm brothers?*" (for which two distinct answers shall be returned).

Precision is defined as the number of correctly answered questions, divided by the number of returned answers, i.e.,

$$Precision = \frac{\text{\#Correct distinct answers given by the system}}{\text{\#Total number of answers given by the system}} . \tag{2.19}$$

Recall is defined as the number of correctly answered questions, divided by the number of possible

Table 2.5: Example of the reciprocal ranks for a test set of questions, with correct answers shown in **bold**.

Question	Ranked list of answers	Rank	Reciprocal Rank
What is the capital of Portugal?	**Lisbon**, Coimbra, Oporto	1	1
When was Mozart born?	1884, 1870, **1756**	3	1/3
Who is the 44th president of the USA?	Bill Clinton, **Barack Obama**, Ron Paul	2	1/2

correct answers, i.e.,

$$Recall = \frac{\text{\#Correct distinct answers given by the system}}{\text{\#Total number of possible correct answers}} . \tag{2.20}$$

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

$$F_\beta = \frac{(\beta^2 + 1) \times Precision \times Recall}{\beta^2 \times Precision + Recall} , \tag{2.21}$$

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 $\beta < 1$ favours precision. When precision and recall are equally weighted (i.e., $\beta = 1$), the measure usually goes by the name of F_1.

As an example, given the question "*List all the states in the USA.*", a system that returned 35 correct answers (out of 50), and 5 incorrect ones, would have a precision of $\frac{35}{40} = 87.5\%$, a recall of $\frac{35}{50} = 70.0\%$, and $F_1 = \frac{2 \times 0.875 \times 0.7}{0.875 + 0.7} \simeq 78.0\%$.

2.4.3 Mean reciprocal rank

The Mean Reciprocal Rank (**MRR**) is a specifically tailored evaluation measure for question answering, originally introduced by (Voorhees, 1999) in the first edition of the QA track of TREC. It can be used to evaluate systems that return ranked lists of possible answers for a given question. The reciprocal **rank** of an individual question is then defined to be the multiplicative inverse of the rank of the first correct answer, or zero if no answer is returned. Thus, the mean reciprocal rank is the average of the reciprocal rank of every question in a test set. More formally, for a test set of N questions, the MRR is given by:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} . \tag{2.22}$$

Table 2.5 depicts a fictitious run of a system for a small test set of three questions, with $MRR = \frac{1 + \frac{1}{3} + \frac{1}{2}}{3} \simeq 0.61$.

2.5 Conclusions

In this chapter, we gave a general overview of the question answering field, with a special focus on machine learning techniques. We concluded that many machine learning techniques have only been applied to individual tasks within a QA system – such as question classification –, since no evidence was found of a full-blown system that used machine learning in all components. However, the observed results for each different task are very encouraging, suggesting that it is possible to build high performance QA systems based on machine learning methods. Nevertheless, natural language processing techniques will always be of vital importance for QA, since many of them are still required for the extraction of features that are commonly used in the training phase of many learning algorithms.

3 Question classification

The present chapter describes the machine learning-based question classifier of the QAML system, as well as its empirical evaluation. Section 3.1 briefly reviews the question classification task, and informally describes this thesis' approach to it. Section 3.2 presents the question type taxonomy that was employed throughout this work. Section 3.3 examines the set of features that were used in the question classification experiments. Section 3.4 describes the evaluation methodology that was followed to conduct the experiments. Section 3.5 details the experiments, reports the results that were attained in each, and compares them to other results reported in the literature. Finally, in Section 3.6, the results are discussed, and suggestions for improvements are provided.

3.1 Introduction

Question classification is a very important step in a question answering system. Its goal is to, given a question posed in natural language, assign a semantic category to that question, which represents the type of answer that is being sought after. The set of question categories into which the questions are to be assigned is referred to as the question type taxonomy. For instance, using the taxonomy adopted in this work – described in the following section –, the question "*Which country borders Portugal?*" ought to be classified into the category LOCATION:COUNTRY.

The selected question category can then be used for two different purposes. First, it can be used to narrow down the number of possible answer candidates, as we shall see in more detail in Chapter 5, when we introduce the answer extraction module of our system. For example, knowing that a question belongs to the category CITY, allows the system to only consider cities as possible answers. Second, depending on the question category, different processing strategies can be chosen to find an answer. For instance, if a question is assigned to the category DEFINITION, the system can search for possible answers in encyclopaedic sources – such as Wikipedia –, which are more likely to contain an accurate answer. These strategies will be covered to a greater extent in Chapter 4.

Furthermore, since it is the first task to be executed in our system, a misclassified question in this stage can hinder the ability of the system to reach an accurate answer, because it can cause downstream components to make wrong assumptions about the question. Hence, it is of crucial importance to have a precise question classifier, in order to achieve good overall results.

3.1.1 Our approach

In this work, we adopt a machine learning approach to question classification. More specifically, we model the task of question classification as a supervised learning classification problem, where the goal is to learn a hypothesis that can accurately predict the categories of unseen questions. In order to train the learning model, we designed a rich set of features that are predictive of question categories, which are described in Section 3.3. We then performed a series of different experiments (Section 3.5), with the goal of determining the most discriminative set of features that yield the most accurate results. The experimental methodology used for the experiments is presented in Section 3.4.

3.2 Question type taxonomy

In this thesis, we employ Li and Roth's (X. Li & Roth, 2002) two-layer question type taxonomy, presented in full detail in Table 3.1. It consists of 6 coarse grained categories (ABBREVIATION, DESCRIPTION, ENTITY, HUMAN, LOCATION, and NUMERIC), which are further divided into 50 finer grained categories. We also assume that a question can only belong to a single category.

In our experiments, we use both coarse and fine grained categories for the evaluation of the question classifier. However, in later stages of the question answering pipeline, only the finer grained categories are utilised, since they provide stronger constraints on the possible answers than coarse grained categories. For example, given the question "*When did World War II began?*", it is much more useful to know that the type of answer that we seek is a DATE, than merely to know that it is a NUMERIC type.

3.3 Question feature set

One of the main challenges in developing a supervised classifier for a particular domain is to identify and design a rich set of features – a process which is generally referred to as *feature engineering*. In the subsections that follow, we present the different types of features that were used in the question classifier, and how they are extracted from a given question.

3.3.1 Lexical features

Lexical features refer to *word related* features that are extracted directly from the question. In this work, we use word level n-grams as lexical features. We also include in this section the techniques of stemming and stopword removal, which can be used to reduce the dimensionality of the feature set.

Table 3.1: Li&Roth's question type taxonomy, with example questions for each category.

Coarse	Fine	Example
ABBREVIATION	abbreviation	What is the abbreviation of General Motors?
	expansion	What does the abbreviation AIDS stand for?
DESCRIPTION	definition	What is ethology?
	description	What is the origin of the name Katie?
	manner	How do you get to the top of the Eiffel Tower?
	reason	Why do men snore?
ENTITY	animal	What is the fastest fish in the world?
	body	What's the colored part of the eye called?
	color	What color is Mr. Spock's blood?
	creative	Name a novel written by John Steinbeck.
	currency	What currency is used in Australia?
	medical disease	What is the fear of cockroaches called?
	event	What are the historical trials following World War II called?
	food	What is the world 's best selling cookie?
	instrument	What instrument is Ray Charles best known for playing?
	language	What language is mostly spoken in Brazil?
	letter	What letter adorns the flag of Rwanda?
	other	What's the highest hand in straight poker?
	plant	What is the state tree of Nebraska?
	product	What is the best brand for a laptop computer?
	religion	What religion has the most members?
	sport	What game is Garry Kasparov really good at?
	substance	What is glass made of?
	symbol	What playing card symbolizes death?
	technique	What basketball maneuver did Bert Loomis invent?
	term	What was another name for East Germany?
	vehicle	What was the name of Captain Bligh's ship?
	word	What English word contains the most letters?
HUMAN	description	Who was Confucius?
	group	Which company created the Internet browser Mosaic?
	individual	Who invented baseball?
	title	What is Nicholas Cage's occupation?
LOCATION	city	What is the capital of Burkina Faso?
	country	Name the largest country in South America.
	mountain	What is the highest mountain in the world?
	other	What continent is Bolivia on?
	state	What U.S. state is Fort Knox in?
NUMERIC	code	What is the airport code for Los Angeles International?
	count	How many calories are there in a glass of water?
	date	When was Beethoven born?
	distance	How tall is the giraffe?
	money	How much did the first Barbie doll sell for in 1959?
	order	What chapter of the Bible has the most verses?
	other	What is the frequency of VHF?
	percent	What are the odds of giving birth to twins?
	period	How old is the sun?
	speed	How fast is light?
	temperature	What is the temperature today?
	size	How big is a quart?
	weight	What is the average weight for a man?

3.3.1.1 Word level n-grams

A word level n-gram is a sequence of n consecutive words from a given question. In this work, we have experimented three different values for n, namely, $n = \{1, 2, 3\}$, which are hereafter referred to as unigram, bigram, and trigram, respectively. Each word level n-gram is used as a binary feature, indicating whether a question contains the n-gram or not. The following exemplifies unigrams, bigrams, and trigrams for the question "*Where was Kennedy born?*":

- **Unigrams**: *Where*, *was*, *Kennedy*, *born*, *?*;
- **Bigrams**: *Where was*, *was Kennedy*, *Kennedy born*, *born ?*;
- **Trigrams**: *Where was Kennedy*, *was Kennedy born*, *Kennedy born ?*.

The rationale behind this feature is that questions of the same category tend to share word n-grams. For instance, the unigram *city* appears often in questions of type LOCATION:CITY, which can be a good indicator that the question belongs to this category. Another example is the bigram *Who was* which tends to appear associated with questions of type HUMAN:DESCRIPTION.

3.3.1.2 Stemming and Stopword removal

Stemming is a technique that reduces words to their grammatical roots or *stems*, by removing their affixes. For instance, after applying stemming, the words *inventing* and *invented* both become *invent*. We exploit this technique in our question classifier in the following manner. First, we represent the question using the bag-of-words model as previously described. Second, we apply Porter's stemming algorithm (Porter, 1980) to transform each word into its stem. The following two examples depict a question before and after stemming is applied, respectively.

(3.1) Which countri***es*** are border***ed*** by Franc***e***?

(3.2) Which countri are border by Franc?

Another related technique is to remove stop words, which are frequently occurring words with no semantic value, such as the articles *the* and *an*. Both of these techniques are mainly used to reduce the feature space of the classifier – i.e., the number of total features that need to be considered. This is achieved by collapsing several different forms of the same word into one distinct term by applying stemming; or by eliminating words which are likely to be present in most questions – stopwords –, and which do not provide useful information for the classifier.

3.3.2 Syntactic features

Syntactic features denote *syntax related* features, that require an analysis of the grammatical structure of the question to be extracted. This class of features include the question headword in Section 3.3.2.1, and part-of-speech tags in Section 3.3.2.2.

3.3.2.1 Question headword

The question headword [1] is a word in a given question that *represents* the information that is being sought after. In the following examples, the headword is in bold face:

(3.3) What is Australia's national **flower**?

(3.4) Name an American made **motorcycle**.

(3.5) Which **country** are Godiva chocolates from?

(3.6) What is the name of the highest **mountain** in Africa?

In Example 3.3, the headword *flower* provides the classifier with an important clue to correctly classify the question to ENTITY:PLANT. By the same token, *motorcycle* in Example 3.4 renders hints that help classify the question to ENTITY:VEHICLE. Indeed, all of the aforementioned examples' headword serve as an important feature to unveil the question's category, which is why we dedicate a great effort to its accurate extraction.

Headword extraction

The approach followed in this thesis for the extraction of the question headword requires a parse tree of the question. We use the Berkeley Parser (Petrov & Klein, 2007) for the purpose of parsing questions, trained [2] on the QuestionBank (Judge et al., 2006), a treebank of 4000 parse-annotated questions. Figure 3.1 shows the parse tree [3] of Examples 3.3 and 3.4.

The resulting parse tree of a question is then traversed top-down to find the question headword, using Algorithm 2. The algorithm works by determining the head Y_h of a non-terminal X with production rule $X \rightarrow Y_i \cdots Y_n$, using head-rules which decide which of the $Y_i \cdots Y_n$ is the head (APPLY-RULES).

[1] In the literature, there isn't an agreement on what exactly a question headword is. Some authors utilise the term *informer spans* (Krishnan et al., 2005) to refer to it, although their definition is different from the one used in this work, which is more similar to the one described in (Huang et al., 2008).

[2] Initially, we used a model trained on the *standard* Wall Street Journal (WSJ) corpus, but we found that many questions were being incorrectly parsed, which limited the ability to accurately extract the headword. This was somewhat expected, as the WSJ is a journalistic news corpora, and does not contain many natural language questions.

[3] Punctuation is omitted for the sake of simplicity.

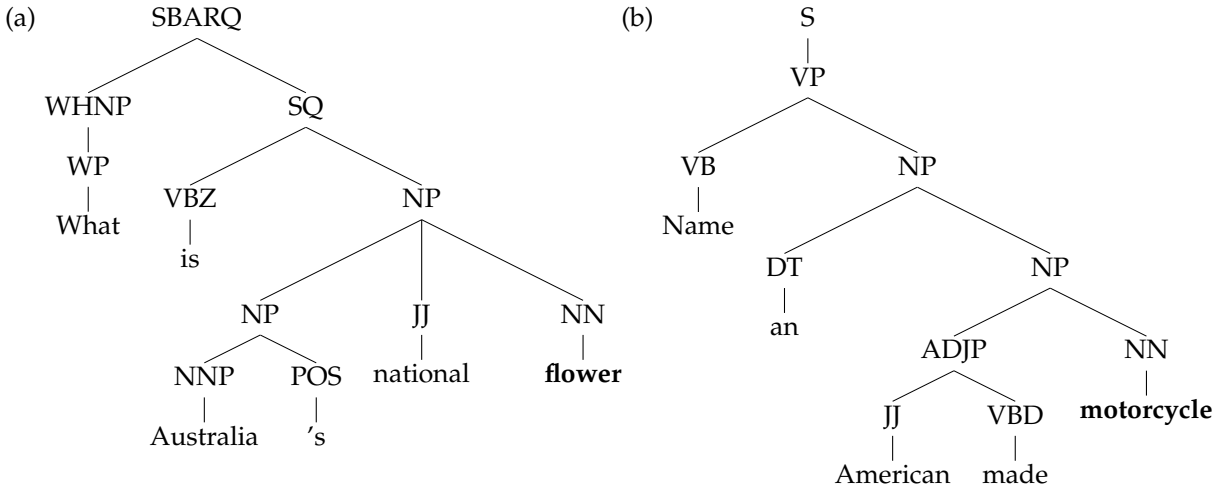

Figure 3.1: (a) Parse tree of the question *What is Australia's national flower?*. (b) Parse tree of the question *Name an American made motorcycle.*. The question headword is in bold face.

This process is then repeated recursively from Y_h, until a terminal node is reached. The head-rules used in this work are a heavily modified version of those given in (M. J. Collins, 1999), specifically tailored to extract headwords from questions. A subset of these rules is listed in Table 3.2.

Algorithm 2 Question headword extraction algorithm

```
procedure EXTRACT-QUESTION-HEADWORD(tree, rules)
    if TERMINAL?(tree) then
        return tree                                   ▷ Returns the headword
    else
        child ← APPLY-RULES(tree, rules)              ▷ Determine which child of tree is the head
        return EXTRACT-QUESTION-HEADWORD(child, rules)
    end if
end procedure
```

As an example of how the algorithm works, consider the parse tree of Figure 3.1a. We start by running the root production rule *SBARQ* → *WHNP SQ* through the head-rules, which specify that the search is conducted from left to right, to find the first child which is an *SQ*, *S*, *SINV*, *SBARQ*, or *FRAG*, thereby identifying *SQ*. A similar reasoning can be applied to determine *NP* as the head of *SQ*. Having the production rule *NP* → *NP JJ NN*, we search from right to left *by position* [4], i.e., starting with the child at the right-most position, we first test it against *NP* – which doesn't match – and then against *NN*, which yields a match. At last, since *NN* is a pre-terminal, the terminal *flower* is returned as the headword.

There are, however, a few exceptions to the head-rules. Consider the parse tree of Example 3.5, depicted in Figure 3.2. In this example, if we follow the aforesaid head-rules, *chocolates* would be extracted as the headword, instead of *country* which is the correct headword. This happens because the rule for

[4]Note that if the direction of *NP* was *Right* by category instead of by position, the left *NP* would be chosen, as we would have searched first by category and then by position, i.e., we would have checked first if there was any *NP* in the entire right-hand-side of the rule, rather than testing it against the child at the right-most position.

Parent	Direction	Priority List
S	Left	VP S FRAG SBAR ADJP
SBARQ	Left	SQ S SINV SBARQ FRAG
SQ	Left	NP VP SQ
NP	Right *by position*	NP NN NNP NNPS NNS NX
PP	Left	WHNP NP WHADVP SBAR
WHNP	Left	NP
WHPP	Right	WHNP WHADVP NP SBAR

Table 3.2: Subset of the head-rules used to determine the question headword, also known as a head percolation table. *Parent* is the non-terminal on the left-hand-side of a production rule. *Direction* specifies whether to search from the left or right end of the rule, either by category first and then by position (by default), or vice-versa when explicitly mentioned. *Priority List* presents the right-hand-side categories ordered by priorities, with the highest priority on the left.

SBARQ chooses *SQ* over *WHNP*, where *country* lies in. A possible solution would to be to prioritise *WHNP* over *SBARQ*, but this would cause wrong headwords to be extracted in other examples, such as in the parse tree of Figure 3.1a, which would return *What*.

Clearly, this problem cannot be solved by modifying the head percolation table, which is why we created a set of what we shall refer to as *non-trivial rules*, which determine the correct headword for situations that cannot be covered by the head percolation table. These rules are described below:

1. When *SBARQ* has a *WHXP* [5] child with at least two children, we return *WHXP* and proceed with Algorithm 2. This situation occurs in Figure 3.2a.

2. After applying the previous rule, if the resulting head is a *WHNP* containing an *NP* that ends with a possessive pronoun (*POS*), we return the *NP* as the head. Without this rule, the headword for the example in Figure 3.2b would be *capital* instead of *country*. Note that this rule only applies inside a *WHNP*, allowing the algorithm to correctly extract *birthday* as the headword of the question "What [$_{SQ}$ is [$_{NP}$ Martin Luther King [$_{POS}$'s]] [$_{NN}$ **birthday**]]?" using the head-rules.

3. There are some headwords, such as *type* and *kind*, which serve merely as a *reference* to the real headword. Figures 3.2c and 3.2d illustrate this situation. For instance, the extracted headword of Figure 3.2d is *kind*, which is referring to a kind of *animal*. In this situation, *animal* would be a much more informative headword. To fix this problem, if the extracted headword is either *name*, *kind*, *type*, *part*, *genre* or *group*, and its sibling node is a prepositional phrase *PP*, we use *PP* as the head and proceed with the algorithm. Note that this rule doesn't apply if there isn't a sibling *PP*, as in "*What is Mao's second name?*".

[5] *WHXP* refers to a *Wh*-phrase: *WHNP*, *WHPP*, *WHADJP* or *WHADVP*.

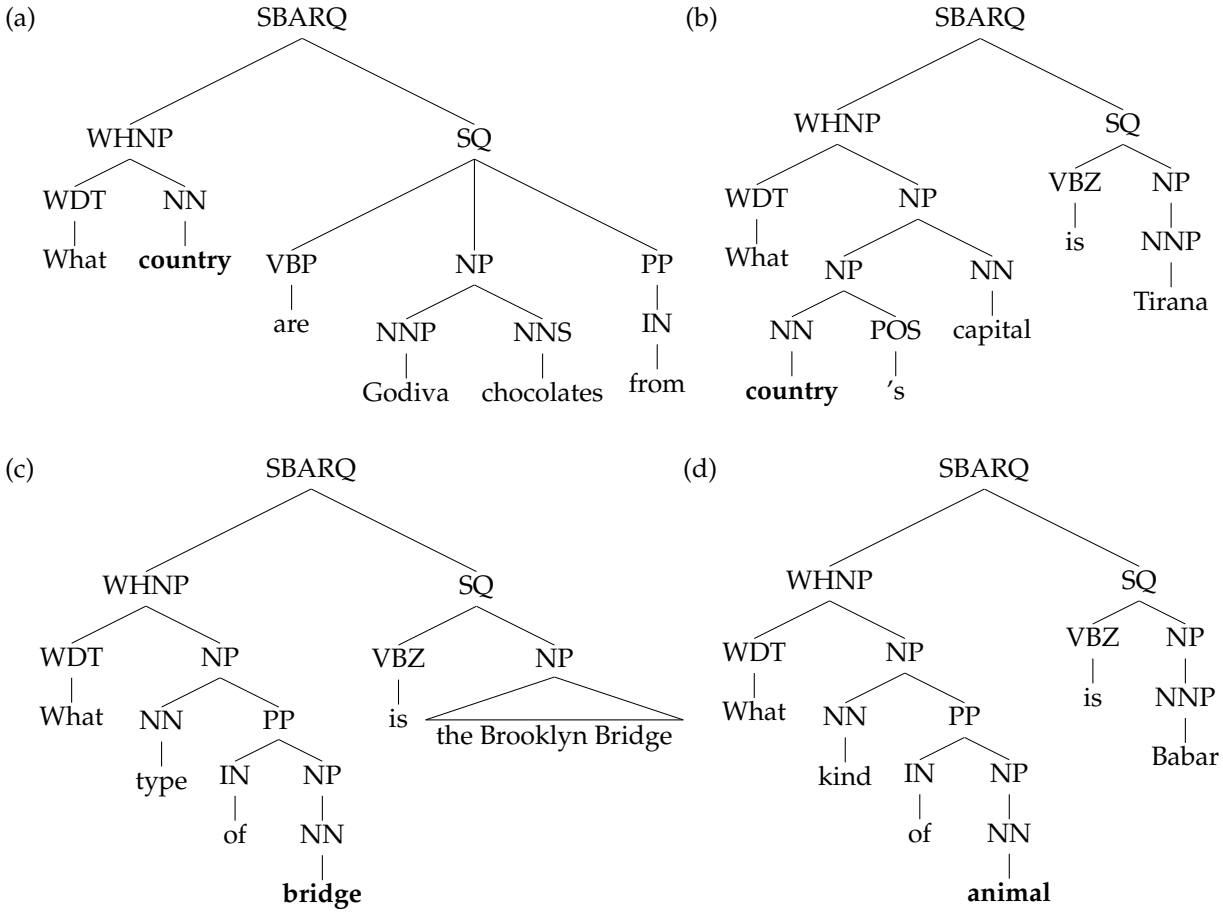

Figure 3.2: Examples of parse trees for which headword extraction is non-trivial. (a) Requires the use of a non-trivial rule for *SBARQ*. (b) Requires two non-trivial rules, the first for *SBARQ* and the second for *WHNP*. Trees (c) and (d) both require a post operation fix.

Algorithm 2 still works in the same manner, the only exception being that the APPLY-RULES procedure now takes the non-trivial rules into account to determine the head of a non-terminal node.

Based on a detailed study of *Wh*-clauses in English by (Trotta, 2000), we theorise that the non-trivial rules, along with the head percolation table, should cover almost all properly formulated questions for English, including questions that do not start with a *Wh*-word, such as "*Jackson Pollack is of what nationality?*".

Question patterns

Nevertheless, there are still some (albeit few) question categories for which our definition of headword doesn't help classification. For instance, in DESCRIPTION:DEFINITION questions such as *What is a bird?*, the headword *bird* is futile because the question is asking for a definition. Moreover, it can mislead the classifier into assigning the category ENTITY:ANIMAL to it. Also, in questions that begin with the *Wh*-word *Who*, *Where* or *When*, the *Wh*-word itself represents the information that is missing, thus func-

tioning by themselves as the question headword.

To prevent some of these pitfalls, we compile a small set of patterns (some of which are adapted from (Huang et al., 2008)), so that when a question matches one of the patterns, a placeholder is returned instead of the question headword. A simplified version of the patterns is presented in Table 3.3.

Placeholder	Question pattern description	Example
P#EXPANSION	begins with *What do(es)* and ends with an acronym – i.e., a sequence of capital letters possibly intervened by dots –, followed by *stands for/mean*; begins with *What is/are* and ends with an acronym	What does AIDS mean ? What is F.B.I. ?
P#DEFINITION	begins with *What is/are* and is followed by an optional determiner and a sequence of nouns	What is ethology ?
P#TERM	begins with *What do you call*	What do you call a shoemaker ?
P#SUBSTANCE	begins with *What is/are* and ends with composed/made of	What is glass made of ?
P#REASON	begins with *What causes*	What causes asthma ?
P#HUM:DESC	begins with *Who is/was* and is followed by a proper noun	Who was Mozart ?
P#WHO	begins with *Who*	Who invented Monopoly ?
P#WHERE	begins with *Where*	Where is France ?
P#WHY	begins with *Why*	Why is the sky blue ?

Table 3.3: Question patterns to avoid extracting a headword, when it is not needed.

The patterns were designed with an emphasis on precision for the most common scenarios, while slightly neglecting their recall. The reason for this was to make sure that if a pattern matches a question, than the question *definitely* doesn't require headword extraction. On the other hand, if a question that doesn't need a headword is not matched by the patterns – e.g., *What is bangers and mash?* –, then this poses no major problem, since the classifier is able to deal with noisy information introduced by a wrong headword, and still devise the correct question category. Algorithm 3 summarizes the entire process of extracting the headword feature.

Algorithm 3 Question headword feature extraction algorithm

```
procedure HEADWORD-FEATURE(question)
    if PATTERN-MATCHES?(question) then
        return placeholder                ▷ Returns the placeholder defined in Table 3.3
    else
        return EXTRACT-QUESTION-HEADWORD(question.tree, rules)        ▷ Algorithm 2
    end if
end procedure
```

3.3.2.2 Part-of-speech tags

Given the parse tree of a question, we extract the pre-terminal nodes to use as features. These nodes represent the part-of-speech (POS) tags or grammatical classes of the question tokens. For example, the POS tags of the question "*What is the capital of France?*" are: *WP*, *VBD*, *DT*, *NN*, *IN*, and *NNP*.

3.3.3 Semantic features

Semantic features introduce semantics or *meaning* to the words in the question. This class includes a semantically enriched version of the headword, and named entities. Named entities and semantically enriched version of the headword pertain to this class of features.

3.3.3.1 Named entities

Named entity recognition is the task of finding and classifying particular names in a natural language text into predefined categories, such as person names, location names, and organization names. For example, given the question "*When did Christopher Columbus discovered Hispaniola?*", a named entity recognizer would (ideally) identify the following named entities (NE)[6]:

(3.7) When did [$_{\text{NE\_PERSON}}$ Christopher Columbus] discovered [$_{\text{NE\_LOCATION}}$ Hispaniola] ?

In this work, we experiment the use of named entities for question classification in two different manners: feature enrichment and feature reduction. In the former, we simply add the recognized named entities' categories as features for the classifier. In the latter case, we substitute the identified named entities with the respective named entity category. For example, the question "*Who is [*$_{\text{NE\_PERSON}}$*Tom Cruise]?*" would become "*Who is* NE_PERSON*?*". The rationale behind this latter approach is that these named entities do not provide much relevant information to the classifier (other than, say, a particular name of a person is present), and can then be conveniently grouped together into a single token – the named entity category.

To extract the named entities from the questions, we employ a Hidden Markov Model based named entity recognizer, provided by `LingPipe` (Alias-i, 2008). Also, only three types of named entities are extracted: NE_PERSON, NE_LOCATION, and NE_ORGANIZATION.

3.3.3.2 Semantic headword

In Section 3.3.2.1, we introduced the *question headword* feature, and stressed its importance to question classification. We now propose and discuss a further potential improvement of the feature, which makes

[6] In order to distinguish named entity categories from question categories, the former are prefixed with "NE_".

use of WordNet (Fellbaum, 1998) to enrich the question headword with semantics. But before delving any further into this subject, consider the following example questions, with their corresponding headword in bold face:

(3.8) What **explorer** was nicknamed Iberia's Pilot ?

(3.9) What **actor** first portrayed James Bond ?

(3.10) What **dictator** has the nickname "El Maximo" ?

Even though all of the above examples fall under the HUMAN:INDIVIDUAL category, the question headword is different in all of them, which limits the usefulness of the feature. The headwords do, however, share a common trait: they are all subordinates (hyponyms) of the word *person*, that is to say, they are all more specific senses of *person* – the superordinate (hypernym). Knowing this information would be useful to accurately classify the previous questions.

We exploited this observation to devise a new feature that adds semantics to the question headword, by using WordNet's lexical hierarchy to associate the headword with a higher-level semantic concept, which represents a question category. This was accomplished as follows. First, sets of related WordNet synsets were manually grouped together into fifty clusters, with each cluster representing a fine-grained question category. Some of these clusters are shown in Table 3.4. Secondly, given a question headword, we translate it into a WordNet synset using a set of heuristics, which we shall describe in a moment. Finally, and since WordNet can be seen as a directed acyclic graph, with synsets as vertices and lexical relations – such as hypernymy – as edges, we employ a breadth-first search on the translated synset's hypernym tree, in order to find a synset that pertains to any of the pre-defined clusters. Algorithm 4 details this process.

Algorithm 4 Semantic headword feature extraction algorithm

```
procedure SEMANTIC-HEADWORD-FEATURE(headword, groups : associative array synset → group)
    root-synset ← TRANSLATE-TO-SYNSET(headword)
    ENQUEUE(root-synset, queue)
    while ¬EMPTY?(queue) do
        synset ← DEQUEUE(queue)
        if synset is in groups then
            return groups[synset]
        else
            for each hypernym in DIRECT-HYPERNYMS-FOR(synset) do
                ENQUEUE(hypernym, queue)
            end for
        end if
    end while
    return not found
end procedure
```

Cluster name	Synsets	Example hyponyms
ENTITY:ANIMAL	animal, animate_being, beast, brute, creature, fauna animal_group	mammal, fish, cat flock, herd, breed
ENTITY:CREATIVE	show music writing, written material, piece of writing	movie, film, tv show song, tune, hymn book, poem, novel
ENTITY:PLANT	vegetation, flora, botany plant, flora, plant life	forest, garden flower, tree, shrub
HUMAN:INDIVIDUAL	person, individual, someone, somebody, mortal spiritual being, supernatural being homo, man, human being, human	actor, leader god, angel, spirit Homo sapiens
HUMAN:GROUP	organization, organisation group, grouping	enterprise, company team, tribe, dynasty
NUMERIC:DATE	date, day of the month	birthday, natal day
NUMERIC:DISTANCE	distance dimension	altitude, elevation width, length, height

Table 3.4: Examples of clusters that aggregate similar synsets together. The complete list can be found in Annex A.

As an illustration of how the algorithm works, consider the headword *actor* of Example 3.9 and its hypernym tree depicted in Figure 3.3. The first three levels of the tree yield no results, since no cluster contains any of the synsets at these levels. At the fourth level of the hypernym chain, however, a member of the HUMAN:INDIVIDUAL cluster is found – *person* –, and thereby the algorithm terminates, returning the cluster's name to use as a feature.

Sense 1
actor, histrion, player, thespian, role player -- (a theatrical performer)
⇒ performer, performing artist
⇒ entertainer
⇒ **person, individual, someone, somebody, mortal, soul**
⇒ organism, being
⇒ living thing, animate thing
⇒ whole, unit
⇒ object, physical object
⇒ physical entity
⇒ entity

Figure 3.3: Hypernym tree for the first sense of the synset *actor*.

This feature is of prime importance to question classification – as we shall demonstrate in Section 3.5 –, because it effectively enriches the headword with semantic knowledge. For instance, in the previous example, besides knowing that *actor* is the headword, we now also know that it IS-A *person*, and that it is associated with the HUMAN:INDIVIDUAL category. Also, in questions where the headword is a rather

uncommon word, such as in "*What is the **wingspan** of a condor?*", this feature is even more important, as it would be very difficult to classify the question as NUMERIC:DISTANCE without knowing that *wingspan* IS-A distance.

Synset translation

In order for the semantic headword feature to be effective, it is crucial to translate the headword into the appropriate WordNet synset. This, however, is not a trivial undertaking due, in grand part, to homonymous and polysemous words – i.e., words with the same spelling but different meanings –, which have multiple senses (synsets) in WordNet.

An example of such a word is *capital* which, in WordNet [7], can refer to *wealth in the form of money* (capital^1), *a seat of government* (capital^2), or even an adjective as in *of capital importance* (capital^3). The task of identifying the correct sense for a word is called word sense disambiguation, for which many algorithms have been devised (Jurafsky & Martin, 2008). For instance, in the question "*What is the **capital** of Portugal?*", the correct sense for the headword is capital^2, and failing to identify it as so can introduce noise to the classifier – e.g., selecting capital^1 could mislead the classifier into categorising the question into NUMERIC:MONEY in lieu of LOCATION:CITY, which is the correct. In this work, we utilise the following three heuristics to aid word sense disambiguation:

1. WordNet synsets are organised according to four part-of-speeches (POS): nouns, adjectives, verbs, and adverbs. This allows for some partial disambiguation of the question headword, by converting the headword's Penn-style POS tag into the corresponding WordNet POS, and then using it to retain only those senses that belong to the converted POS. This would eliminate the adjectival sense of *capital* in the aforesaid example.

2. Besides single words, WordNet also contains entries for compound words (Sharada & Girish, 2004), which are a combination of two or more words that constitute a single unit of meaning, such as *mountain range*. These are very useful, because they are often monosemous and hence do not require further disambiguation. In this work, we try to form compound words from the question headword using the following two strategies [8]: (1) headword is combined with nouns and/or adjectives to its left that act as pre-modifiers (e.g., World **Cup**); (2) headword is combined with a prepositional phrase to its right that acts as a post-modifier (e.g., **capital** of Portugal). The resulting combination is then used, if a *valid* compound word was formed (i.e., a word that exists in WordNet).

[7] WordNet includes more senses for the word *capital*; for the sake of brevity, we have selected only three senses to illustrate our examples.

[8] Other forms of compound words, such as hyphenated forms (e.g., anti-inflammatory) and juxtapositions (e.g., keyword), are already present in WordNet as single words.

3. At last, if after applying the previous two heuristics we are still left with more than one sense for the headword, we opt for the first (most frequent) WordNet sense.

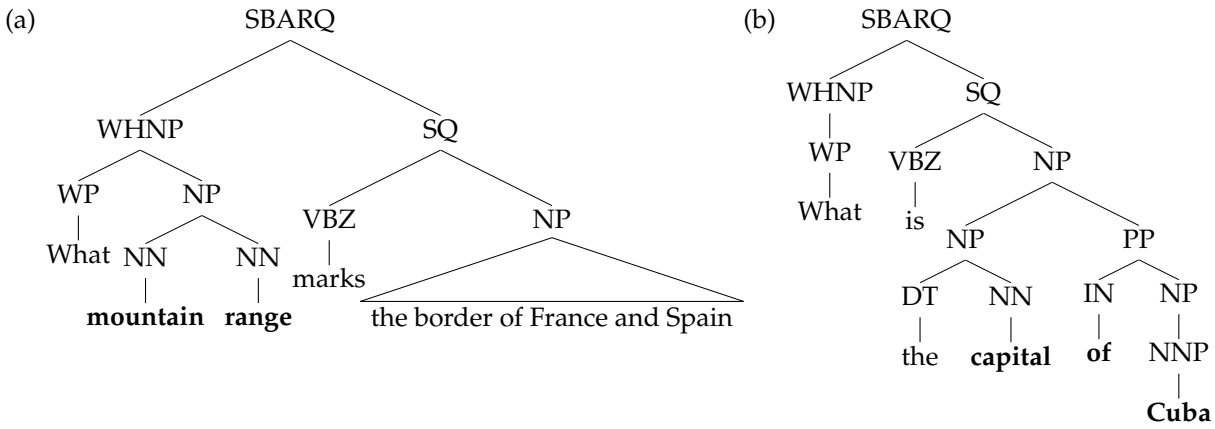

Figure 3.4: Examples of questions for which a compound headword is helpful. (a) The compound headword *mountain range* can help to classify the question into LOCATION:MOUNTAIN, while *range* doesn't. (b) The compound headword *capital of Cuba* does not require disambiguation, as opposed to *capital*.

3.4 Experimental setup

This section describes the experimental setup used for evaluating the performance of the developed question classifier, including the different learning algorithms that were used (Section 3.4.1), the training and test corpus (Section 3.4.2), and the evaluation measures (Section 3.4.3).

3.4.1 Classification algorithms

In our experiments, three different types of supervised classifiers were used: Naïve Bayes (NB), k Nearest Neighbours (kNN), and Support Vector Machine (SVM). A detailed explanation of how NB and SVM work can be found in Chapter 2, Section 2.3.

For all experiments involving SVM, we employed the `LIBSVM` (Chang & Lin, 2001) implementation with a linear kernel, and trained the classifiers using the one-versus-all multi-class strategy, as described in Section 2.3.2 of Chapter 2. As for the the Naïve Bayes and k Nearest Neighbours implementation, we adopted the `LingPipe` (Alias-i, 2008) software package. For kNN, we used the cosine distance as the distance metric, and $k = 1$ – i.e., nearest neighbour search. Every other algorithm was left with its default parameters unchanged, unless otherwise noted.

3.4.2 Data sets

The empirical evaluation of the question classifier was carried out on the publicly available data set of the Cognitive Computing Group at University of Illinois at Urbana-Champaign (UIUC) [9], which consists of a training set of 5500 questions, and a test set with 500 questions. The annotated question categories follow the question type taxonomy described in Section 3.2.

This data set is also one of the most commonly used in the literature for machine learning-based question classifiers (X. Li & Roth, 2002; Zhang & Lee, 2003; Blunsom et al., 2006; Huang et al., 2008), which makes it very suitable for benchmarking purposes by comparing the attained results with other state of the art approaches.

Also, unless otherwise stated, all 5500 questions are used to train the machine learning algorithms, and all 500 questions are used for testing.

3.4.3 Evaluation measures

The evaluation measure used to assess the performance of the question classifier is accuracy – i.e., the fraction of the total number of questions that have been correctly classified. Additionally, the performance of the classifier for each particular class c is evaluated using precision and recall, defined as follows:

$$Precision(c) = \frac{\text{\# of correctly classified questions of category } c}{\text{\# of predicted questions of category } c}\,; \tag{3.11}$$

$$Recall(c) = \frac{\text{\# of correctly classified questions of category } c}{\text{\# of questions of category } c}\,. \tag{3.12}$$

3.5 Experimental results

In the following three subsections (3.5.1-3.5.3), we describe the experiments that were carried out with the goal of determining the most discriminative set of features and settings for the question classifier. We then proceed, in Section 3.5.4, with a comparison of our results with others reported in the literature, and finally, in Section 3.5.5, we discuss the attained results and suggest how they could be improved.

3.5.1 Comparison of classification algorithms

The first experiment was designed to evaluate the individual performance of the three classifiers described in Section 3.4, using simple unigrams as features, and under the coarse grained category. We

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

also investigate the impact of gradually increasing the training set size on the performance of the classifiers, by running five different trials using the first $n = 1000, 2000, 3000, 4000,$ or all 5500 questions of the training set for training. The results are shown in Table 3.5.

	Number of questions in training set				
Algorithm	1000	2000	3000	4000	5500
SVM	78.4%	83.6%	85.0%	86.2%	**88.2%**
Naïve Bayes	70.4%	72.4%	73.0%	75.6%	78.6%
k**NN**	62.6%	68.6%	72.4%	73.0%	75.2%

Table 3.5: Question classification accuracy using different machine learning algorithms and different training set sizes, under the coarse grained category.

The results presented in Table 3.5 demonstrate that SVM outperforms both Naïve Bayes and k Nearest Neighbours by a significant margin in every trial. This was an expected finding, since previous literature on this task – such as (Zhang & Lee, 2003) –, had already reported similar results. As a consequence, the experiments described in the remainder of this chapter were all performed using SVM as the base algorithm, and using the full training set of 5500 questions.

Furthermore, our findings also indicate that training a SVM classifier on 4000 questions results in an accuracy of 86.2% as against 88.2% for training on all 5500 questions, which implies that there is still room for improvement by augmenting the training set. However, since the increase of accuracy from adding 1500 questions to the training set is just 2%, we conclude that the gain in accuracy that could be reached from adding more annotated questions does not justify the effort. Hence, in order to improve the achieved results, we have decided to focus our efforts in using more sophisticated features, rather than increasing the training corpus.

3.5.2 Lexico-syntactic features contribution

In the second experiment, we trained a SVM classifier using different combinations of both lexical and syntactic features, in order to assess their individual and combined contribution to question classification. The experimental results for coarse- and fine-grained classification are shown in Figure 3.5 and Figure 3.6, respectively.

When considering lexical features solely, we can observe that the features that yield the most accurate results are unigrams and bigrams combined, with an accuracy of 90.4% and 81.2% for coarse- and fine-grained classification, respectively. Moreover, we can also conclude that trigrams do not help question classification, as the combination of trigrams with unigrams, bigrams, or both, results in a drop of accuracy, when compared to using these features in isolation.

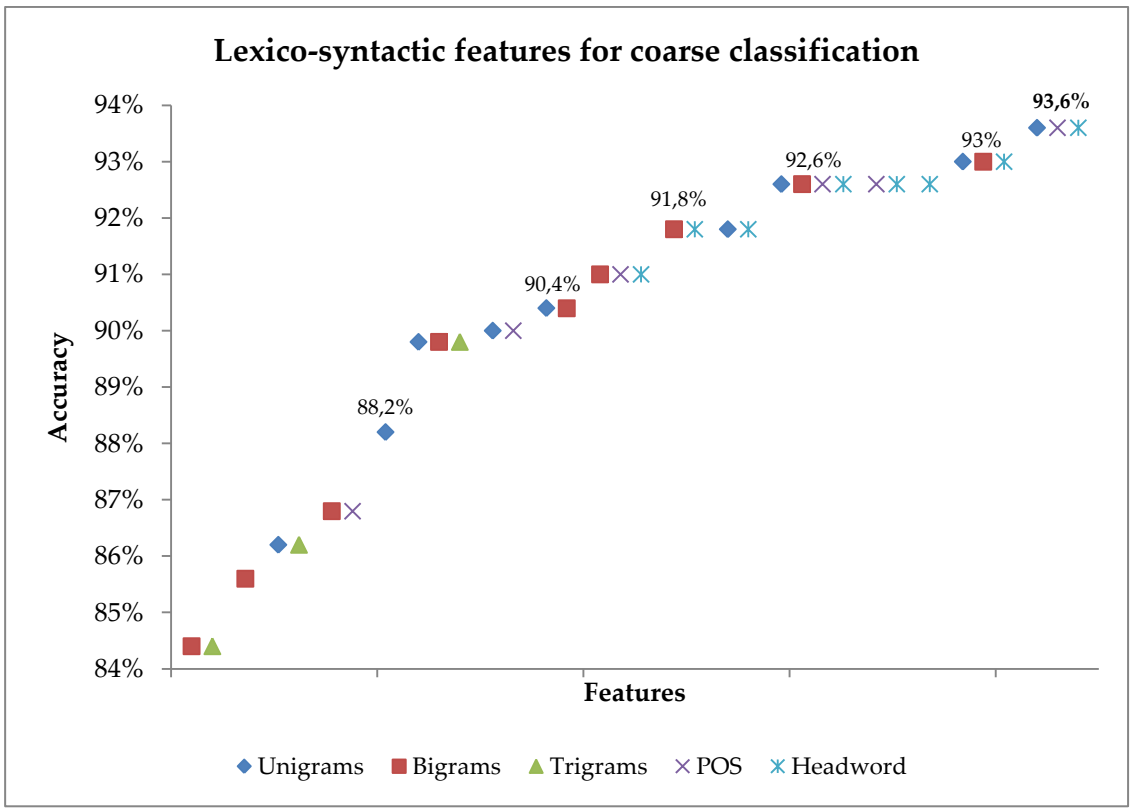

Figure 3.5: Question classification accuracy using different combinations of lexical and syntactic features, under the coarse-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features.

Regarding syntactic features, the question headword has proved to be of great importance to question classification – as we had already predicted –, achieving just by itself 92.6% and 82.8% accuracy for coarse- and fine-grained classification, respectively. The POS tags, however, though informative for coarse-grained classification, are prejudicial for fine-grained classification, which contradicts the results obtained by (F. Li et al., 2008).

Another interesting observation is that bigrams do not provide more information than unigrams do, when combined with the question headword. Indeed, for fine-grained classification, unigrams and the headword feature combined result in 85.4% accuracy, but when adding bigrams to the mix, accuracy drops to 83.6%.

In sum, we can conclude that the most prominent and discriminative lexico-syntactic features are the question headword and unigrams. In the experiment that follows, we experiment the use of these two features in combination with semantic features.

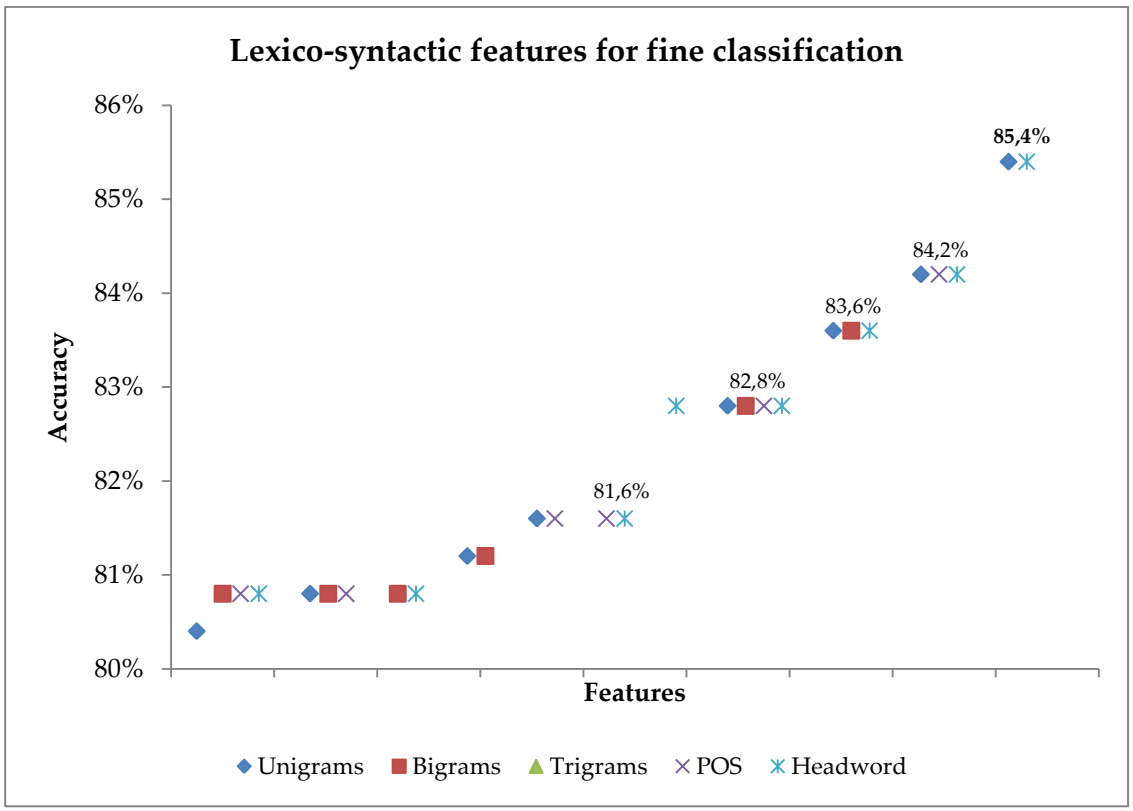

Figure 3.6: Question classification accuracy using different combinations of lexical and syntactic features, under the fine-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features.

3.5.3 Semantic features contribution

The third and final experiment was designed to measure the contribution of semantic features to question classification. Specifically, we experimented different combinations of semantic features with the lexico-syntactic features that yielded the most informative results in the previous experiment. The results for coarse- and fine-grained classification are respectively pictured in Figure 3.7 and Figure 3.8.

The best accuracy attained in this experiment for both coarse- and fine-grained classification – 95.4% and 90.6%, respectively –, is achieved using the combination of the question headword, semantic headword (referred to as WordNet in the figures), and unigrams. The results also reflect the fact that semantic features are more helpful for fine-grained classification – as other authors have already reported (F. Li et al., 2008) –, as these features tend to be most discriminative when there are more classes, and therefore more ambiguity.

With respect to named entities, the experimental results show that, although they improve classification accuracy when combined with lexico-syntactic features, they do not help when combined with the semantic headword. Indeed, when combined with unigrams, question headword, and semantic headword, they are detrimental to performance, resulting in a small drop of accuracy of 0.2% for both

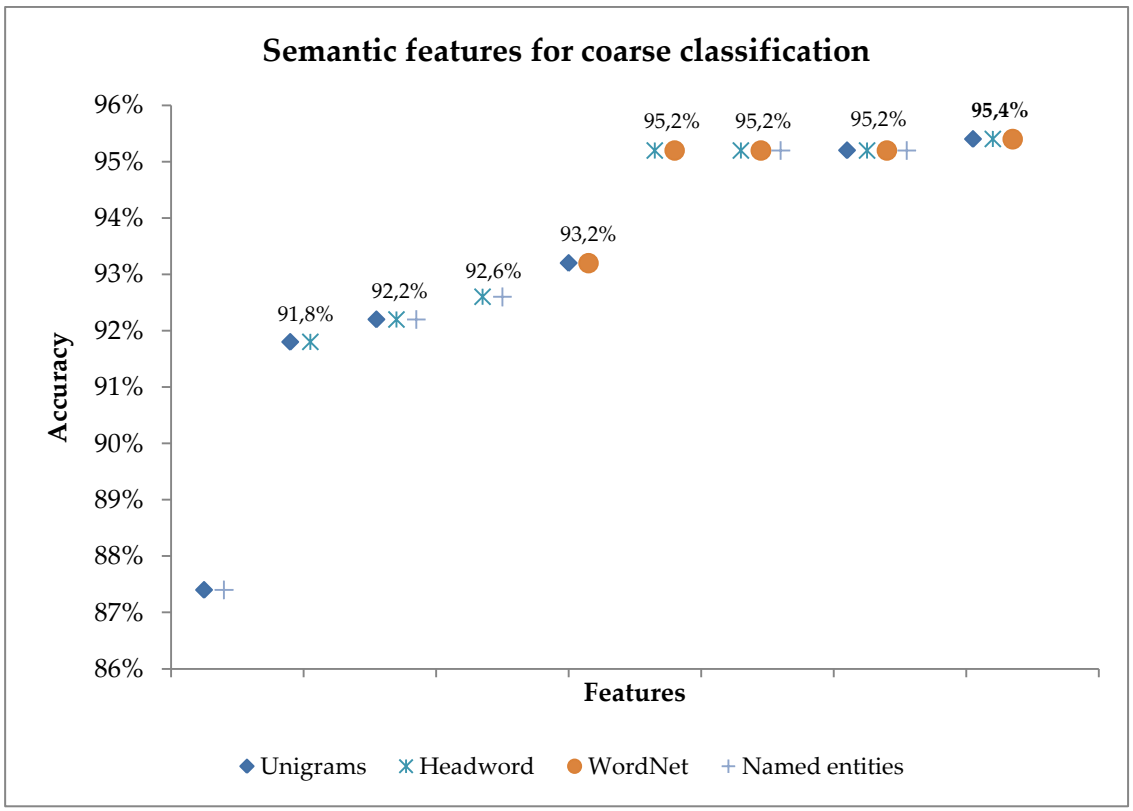

Figure 3.7: Question classification accuracy using different combinations of lexical, syntactic, and semantic features, under the coarse-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features.

coarse- an fine-grained classification. This was an unexpected result, as other works in the literature (Pan et al., 2008; F. Li et al., 2008) have demonstrated that named entities are indeed helpful for question classification. This disparity can be partly explained by the poor performance of our NER and the scarce number of entity types that it is able to recognise. For instance, the NER of (F. Li et al., 2008) is able to recognise more than thirty named entity types, while ours only recognises three.

3.5.4 Comparison with other works

We now compare our results with others reported in the literature. Table 3.6 summarises the question classification accuracy reached by other relevant works in the literature for this particular task. All works were evaluated using similar settings as this work, with the same question type taxonomy, and the same training and test sets.

Regarding coarse-grained classification, the work of (Pan et al., 2008) reported a similar result to ours, with an accuracy of 94%. They utilise a *semantic tree kernel* described in Chapter 2, with the most striking feature being the semantic classes, which can be seen as a simpler version of our semantic head-

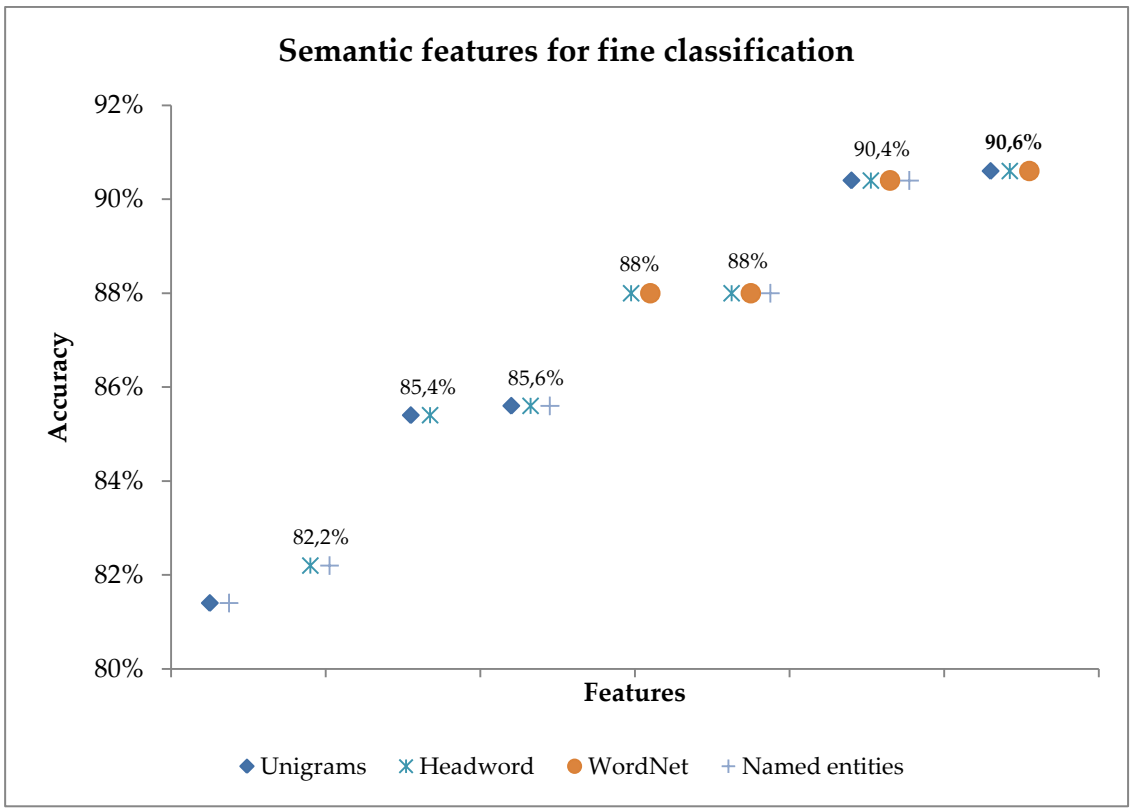

Figure 3.8: Question classification accuracy using different combinations of lexical, syntactic, and semantic features, under the fine-grained category. Juxtaposed symbols represent a combination of the corresponding symbols' features.

word feature. However, in their work, each word in the question is enriched with semantic information, instead of just the question headword, which potentially introduces noise into the classifier, and is probably the reason why our work attains better results.

With respect to fine-grained classification, both the works of (Krishnan et al., 2005) and (Huang et al., 2008) share some similarity to ours, in the sense that they also use the notion of *question headword*, with the main difference being in the techniques that are used to extract the headword from the question, and how WordNet is used to enrich the question headword. For instance, (Krishnan et al., 2005) enriches the feature space with all hypernyms from all senses of the headword, while (Huang et al., 2008) uses all hypernyms at depth ≤ 6.

Furthermore, it is also worth mentioning that our results are achieved using a compact feature space comprising nearly 10,000 distinct features, as against the 200,000 of (X. Li & Roth, 2002) and the 13,697 of (Huang et al., 2008), which results in a very efficient, yet effective question classifier.

	Category granularity	
Author	Coarse	Fine
This work	**95.2%**	**90.6%**
(X. Li & Roth, 2002)	91.0%	84.2%
(Zhang & Lee, 2003)	90.0%	80.2%
(Krishnan et al., 2005)	93.4%	86.2%
(Blunsom et al., 2006)	91.8%	86.6%
(Pan et al., 2008)	94.0%	-
(Huang et al., 2008)	93.6%	89.2%

Table 3.6: Comparison of question classification results attained by this work, against other results reported in the literature.

3.5.5 Discussion

We now provide some further analysis of the achieved results, and discuss how they can be improved.

First, in order to better understand how the question classifier behaves for each fine-grained category, we present in Table 3.7 the precision and recall achieved for each, using the combination of lexical, syntactic, and semantic features that yielded the most accurate results: unigrams, question headword, and semantic headword.

Table 3.7 reflects the fact that some categories are harder to classify than others, as the results are very divergent. For instance, a large source of errors come from the OTHER fine-grained category, of each coarse-grained category. This suggests that it may be helpful to split these broader OTHER-type categories into more specific categories, in order to reduce classification errors.

Nevertheless, we achieved very satisfactory results for question classification, although there is still some margin for improvement. We now discuss some anomalies that we have detected in our experiments, and present some cases where our classifier is unable to correctly classify the question.

While analysing the data sets that were used for training and testing the classifier, we have detected several issues which can affect classifier performance. For instance, some questions contain noisy information such as POS tags – e.g., "*What did 8 , CD NNS VBP TO VB NNP POS NN*" –, which clearly poses a problem to the classifier. Also, the dataset is strongly skewed to certain categories, with the HUMAN:INDIVIDUAL and DESCRIPTION:DEFINITION categories together accounting for more than 25% of the training set, which can hurt classification performance for other less common categories.

Another problem with the data sets is that the labels are not consistent for every question. For instance, the question "*What is the population of Ohio?*" is classified as NUMERIC:OTHER, while "*What is the population in India?*" is classified as NUMERIC:COUNT. Additionally, there are also some questions that are incorrectly classified, such as "*What is the melting point of copper?*" which is classified as NU-

Category	#	Precision	Recall	Category	#	Precision	Recall
ABBREVIATION	9			term	7	85.7%	85.7%
abbreviation	1	100%	100%	vehicle	4	100%	100%
expansion	8	95%	96%	word	0	-	-
DESCRIPTION	138			HUMAN	65		
definition	123	92%	98%	description	3	100%	100%
description	7	75%	86%	group	6	50%	66.7%
manner	2	66.7%	100%	individual	55	94.8%	100%
reason	6	100%	100%	title	1	0%	0%
ENTITY	94			LOCATION	81		
animal	16	93.3%	87.5%	city	18	100%	100%
body	2	100%	100%	country	3	100%	100%
color	10	100%	100%	mountain	3	100%	100%
creative	0	-	-	state	7	100%	100%
currency	6	100%	83.3%	other	50	86.8%	92%
medicine	2	50%	50%	NUMERIC	113		
event	2	50%	50%	code	0	-	-
food	4	100%	100%	count	9	90%	100%
instrument	1	100%	100%	date	47	97.9%	100%
language	2	100%	100%	distance	16	100%	87.5%
letter	0	-	-	money	3	75%	100%
other	12	45.5%	41.7%	order	0	-	-
plant	5	100%	100%	other	12	100%	50%
product	4	100%	25%	percent	3	75%	100%
religion	0	-	-	period	8	70%	87.5%
sport	1	100%	100%	speed	6	100%	100%
substance	15	81.8%	60%	temperature	5	100%	80%
symbol	0	-	-	size	0	-	-
technique	1	100%	100%	weight	4	100%	100%

Table 3.7: Precision and recall for each fine-grained question category, using the feature set of unigrams, question headword, and semantic headword.

MERIC:OTHER instead of NUMERIC:TEMPERATURE. With a thoroughly reviewed data set, we genuinely expect the classification accuracy of our question classifier to be much higher.

Disregarding the issues associated with the data sets, there are still other questions that were not correctly classified for other reasons:

- "*What is the **birthstone** for June?*": Since our lexicon (WordNet) does not contain the headword *birthstone*, our classifier can not determine that it is a substance, and therefore it is unable to classify the question into ENTITY:SUBSTANCE.
- "*In the late 1700's British convicts were used to populate which **colony**?*": In this question, our word sense disambiguation algorithm is unable to identify the correct sense of *colony* as in *a geographical area politically controlled by a distant country*, and instead identifies the sense *a body of people who settle far from home but maintain ties with their homeland*, which causes the classifier to predict the category HUMAN:GROUP instead of LOCATION:OTHER.
- "*What is Valentine's Day?*: For this question, our system predicts NUMERIC:DATE instead of DESCRIPTION:DEFINITION, which is the correct category. However, the question certainly has a degree of ambiguity, as the user could be asking for the specific day at which Valentine's Day occurs, or a description of what is Valentine's Day.

3.6 Conclusions

In this chapter, we presented a machine learning-based approach to question classification, modeled as a supervised learning classification problem. In order to train the learning algorithm, we developed a rich set of lexical, syntactic, and semantic features, among which are the *question headword* and *semantic headword*, which we deemed as crucial for accurate question classification. We then proceeded with a series of experiments to determine the most discriminative set of features, which proved to be the combination of *unigrams*, *question headword*, and the *semantic headword* feature. Using an SVM trained on these features, we attained 95.2% and 90.6% accuracy for coarse- and fine-grained classification, respectively, which, as we write, outperforms every other state-of-the-art result reported in the literature. Furthermore, we also suggested how these results could be improved, by using a better training and test set, and extended question type taxonomy.

4 Passage retrieval

This chapter introduces the passage retrieval component of the QAML system. Section 4.1 reexamines the passage retrieval task, and informally describes this work's approach to it. Section 4.2 lists the information sources from which relevant passages are retrieved. Section 4.3 depicts the query formulation strategies that were employed in this thesis. In Section 4.4, the method that is used to submit formulated queries into suitable information sources is presented. At last, in Section 4.5, conclusions are drawn and directions for future work are provided.

4.1 Introduction

After question classification, the next step in a question answering system is to find relevant passages where the answer to a given question might be found. In QA parlance, this step is usually referred to as *passage retrieval*, and can be further divided into two sub-steps, as follows:

1. **Query formulation**, in which a question is translated into a suitable representation that can be used by an information source to retrieve relevant passages. Note that this representation is not necessarily an unstructured set of keywords – indeed, it can be based on a structured query language, as we shall see in more detail in subsequent sections.
2. **Query submission**, wherein the generated queries from the previous step are fed into the endpoint [1] of one of the information sources presented in Section 4.2, in order to retrieve relevant passages that may contain the answer to a given question.

The retrieved passages are then used by the answer extraction component of the system – described in Chapter 5 –, to extract plausible answers.

4.1.1 Our approach

In this thesis, we employ a multi-strategy approach to passage retrieval, with each strategy tailored to a specific question category or groups of question categories. For instance, for DESCRIPTION:DEFINITION

[1] Endpoint is the entry point to a service, into where queries can be sent.

questions, Wikipedia is used as the information source, whereas for factoid questions, such as the ones that pertain to the HUMAN:INDIVIDUAL category, Google's search engine is used instead.

The information sources utilised in this work are presented in Section 4.2, along with the question categories that each deal with. Furthermore, we propose several query formulation strategies – described in Section 4.3 –, in order to improve the results returned by the information sources. In particular, we shall present a technique that allows the system to learn very precise query formulation patterns, using a semi-supervised algorithm based on bootstrapping. Finally, in Section 4.4, we discuss how the queries are submitted to the information sources and how the results are retrieved.

4.2 Information sources

In the following subsections, we present the information sources that were used in this work to search for relevant passages.

4.2.1 Google search

Google search [2], originally developed by (Brin & Page, 1998), is a multilingual web search engine, indexing over one trillion unique web pages (Google, 2008).

In this work, we use Google search to retrieve passages for factoid-type questions. These passages correspond to the summaries – also called *snippets* in IR jargon – that the search engine associates with each returned search result. The reason we use snippets instead of the content of the corresponding web page is efficiency, as a full-text examination of the web page would be a very expensive task computationally. Moreover, informal experiments have also showed that answers to many factoid-type questions can be found directly in the snippet.

4.2.2 Wikipedia

Wikipedia [3] is a multilingual, web-based, and collaboratively edited encyclopaedia, with over thirteen million articles, as of 2009 (Wikipedia, 2009).

Due to Wikipedia's encyclopaedic nature, we employ it in this work to answer non-factoid questions that require longer answers. In particular, we utilise Wikipedia to answer DESCRIPTION:DEFINITION and HUMAN:DEFINITION questions. Also, for questions whose cardinality is greater than one – i.e., list questions that require more than one answer –, Wikipedia is also used.

[2] http://www.google.com
[3] http://www.wikipedia.org/

4.2.3 DBpedia

DBpedia (Auer et al., 2007) is a *community effort to extract structured information from Wikipedia and to make this information available on the Web*. Put differently, DBpedia can be seen as a machine-readable version of Wikipedia, which provides a way to ask sophisticated queries against Wikipedia's unstructured contents, using declarative query languages, such as SPARQL [4].

DBpedia is used in this work in conjunction with Wikipedia, using a strategy that we shall discuss later in this chapter, when we present the query formulation strategies. 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.

4.3 Query formulation

In the subsections that follow, we examine the query formulation strategies that were used in this work.

4.3.1 Keyword-based

The most simple query formulation strategy consists in generating a query that comprises all the words in a given question, except stop words, such as articles and adverbs. For instance, given the question "*When was Beethoven born ?*", a query with the keywords *Beethoven* and *born* would be generated.

This query formulation strategy is used to generate queries for use in Google search, and therefore applies for every question whose category is associated with the Google search information source – i.e., factoid-type questions.

4.3.2 Wikipedia & DBpedia

As we have already mentioned in Section 4.2, we use both Wikipedia and DBpedia to retrieve answers to DESCRIPTION:DEFINITION and HUMAN:DESCRIPTION questions. We now delve into the details of the query formulation strategies that allows us to do that.

The general idea of this combined strategy is to use Wikipedia's search facilities to locate the title of the article where the answer to a given question might be found, and then use DBpedia to retrieve the *abstract* [5] of the article, which is returned as the answer.

[4] SPARQL Protocol and RDF Query Language.
[5] The abstract of a Wikipedia article roughly corresponds to the first paragraph of the article.

More specifically, this scheme calls for two different query formulation strategies, as follows. First, we formulate the original question into a set of keywords, either by using the strategy presented in the previous section, or by using one of the question headword patterns described in Chapter 3 (if applicable). Second, we use Wikipedia's API [6] to perform a full-text search on Wikipedia for the first relevant article, using the formulated question as input. Finally, and since there is a one-to-one mapping between Wikipedia articles and DBpedia resources, we transform the returned article's title into a DBpedia resource, and use a SPARQL query [7] to retrieve the abstract of the article.

For a better understanding of how this strategy works, consider the question "*Who was Afonso Henriques ?*". First, a query with the term *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 SPARQL query depicted in Figure 4.1 to retrieve the abstract of the article from DBpedia.

```
SELECT ?abstract WHERE {
    <http://dbpedia.org/resource/Afonso_I_of_Portugal> <http://dbpedia.org/property/abstract> ?abstract .
    FILTER (lang(?abstract)="en")
}
```

Figure 4.1: SPARQL query to retrieve the abstract of Wikipedia's *Afonso I of Portugal* article from DBpedia.

4.3.3 Pattern-based

In Chapter 2, we presented several works that discussed how to take advantage of the redundancy available in the Web to find answers. In particular, we examined `AskMSR` (Brill et al., 2002) and `MULDER` (Kwok et al., 2001), whose underlying assumption is based on the observation that, given a large collection of documents – e.g., the Web itself –, the answer to a given question will probably occur in sentences that contain a rewrite of the original question. For example, given the question "*What is the capital of Portugal ?*", a possible rewrite would be "*"the capital of Portugal is"*", which is very likely to appear followed by the answer *Lisbon*.

In this work, we also leverage the redundancy available in the Web, by rewriting the original question in a way that makes it easier to locate answers. However, contrary to the aforementioned works who have created the question rewrite patterns manually, we developed a novel semi-supervised algorithm to extract the patterns. This semi-supervised approach has the advantage of being more easy to adapt and extend with new rules, as opposed to the rule-based approach which requires extra human labour every time new rules are added. Moreover, our approach also allows us to empirically evaluate the precision of each rewrite pattern.

[6] Available at `http://wikipedia.org/w/api.php`.
[7] DBpedia's SPARQL endpoint is available at `http://DBpedia.org/sparql`.

The developed algorithm is based on the bootstrapping technique presented in (Ravichandran & Hovy, 2001), and involves the following two stages. First, we use a question-answer seed pair to bootstrap new question rewrite patterns, using Algorithm 5. Second, the learnt patterns are validated with Algorithm 6, using a different question-answer pair as input.

Algorithm 5 Question rewrite patterns bootstrapping algorithm

```
procedure PATTERN-BOOTSTRAP(seed : question-answer pair)
    patterns ← []
    phrasal-nodes ← GET-PHRASAL-NODES(seed.question.parse-tree)
    for each permutation in PERMUTE({phrasal-nodes, *, pair.answer}) do
        query ← ENCLOSE-DOUBLE-QUOTES(permutation)
        results ← GOOGLE-SEARCH(query)
        for each sentence in results.sentences do
            if MATCHES(sentence, permutation) then
                pattern ← REWRITE-AS-PATTERN(permutation, phrasal-nodes)
                patterns ← ADD(patterns, pattern)
            end if
        end for
    end for
    return patterns
end procedure
```

Algorithm 6 Question rewrite patterns validation algorithm

```
procedure PATTERN-VALIDATION(patterns : patterns to validate, seed : question-answer validation pair)
    threshold ← 0.75
    validated-patterns ← []
    for each pattern in patterns do
        query ← REWRITE-AS-QUERY(seed, pattern)
        query ← ENCLOSE-DOUBLE-QUOTES(query)
        results ← GOOGLE-SEARCH(query)
        precision ← results.size/max-size
        if precision ≥ threshold then
            validated-patterns ← ADD(validated-patterns, pattern)
        end if
    end for
    return validated-patterns
end procedure
```

Regarding the first stage, we start by generating permutations of the set comprising the seed answer, the phrasal nodes of the seed question (excluding the *Wh*-phrase), and a wildcard * which stands as a placeholder for one or more words. The reason we use phrasal nodes instead of question tokens is because they represent a single unit of meaning, and therefore should not be broken down into parts (except for verb phrases). For example, in the question "*What is the capital of Spain ?*", it does not make sense to divide the noun-phrase *the capital of Spain*, as that would lead us to generate many meaningless permutations, such as *Spain of capital the Madrid is*. The wildcard is also very important to add diversity into the generated patterns. For instance, considering the previous question and the sentence *the capital*

of Spain is called Madrid, a wildcard is required to match the verb *called*.

After the permutations have been created, each is enclosed in double quotes and sent to Google search. The double quotes ensure that each search result contains the exact quoted permutation, with every word in the exact same order in which it appears in the original query. The results are then broken down into sentences, and if there exists a sentence that matches the respective permutation, we rewrite it as a pattern. To illustrate this concept, consider again the question "[$_{\text{WHNP}}$ What] [$_{\text{VBZ}}$ is] [$_{\text{NP}}$ the capital of Spain]", and suppose the sentence *the capital of Spain is called Madrid* matches the permutation *the capital of Spain is * Madrid*. The resulting pattern would then be "*NP VBZ* called {*ANSWER*}", which is created by replacing each phrasal node with the respective tag, and the seed answer with the tag "{ANSWER}".

While many of the patterns generated by Algorithm 5 are sufficiently generic to be applied to other questions, there are others who are specific to the seed question-answer pair. For instance, consider the pattern "{*ANSWER*}, with a population of 5561750, *VBZ NP*", extracted from the sentence "Madrid, with a population of 5561750, is the capital of Spain". Clearly, this pattern will only work for the seed question that was used to generate it, and therefore must be ruled-out. To eliminate these *false positives*, we use Algorithm 6, which also requires the use of a question-answer validation pair. Note that, though the bootstrapping and validation question-answer pairs are lexically distinct, they must share the same syntactic structure.

The validation algorithm works by testing each generated pattern against the validation pair, and calculating its precision, which is the ratio between the number of times the pattern correctly matched and the number of times the pattern was expected to match – this is typically the maximum number of results retrieved by the search engine which, in Google search, is eight. For example, using the previously mentioned pattern, and the validation pair "*What is the capital of Russia ?-Moscow*", the query "Moscow, with a population of 5561750, is the capital of Russia" would be issued, resulting in zero results – and thus zero precision, which would cause the pattern to be ruled-out, as the algorithm dictates that each pattern must have precision ≥ 0.75 in order to be retained.

The final validated patterns are then stored for further use, along with a *key* which is a concatenation of the phrasal nodes' tags of the seed question that was used to generate the patterns. Then, given a question, this key is used to check whether or not the question has any applicable patterns.

We now present a worked example to reinforce the understanding of the algorithm. Consider the seed pair "*Where is the Louvre Museum ?-Paris*" and the validation pair "*Where is the CN Tower ?-Toronto*". First, the seed question is parsed, and its phrase nodes – depicted in Figure 4.2 – are extracted. We then permute the set $\{\text{is}, \text{the Louvre Museum}, *, \text{Paris}\}$, which results in $4!$ distinct permutations. Each permutation is then transformed into a query, which is issued to Google search. For instance, consider the query *the Louvre Museum is * Paris*, which results in the pattern "*NP VBZ located in* {*ANSWER*}". The

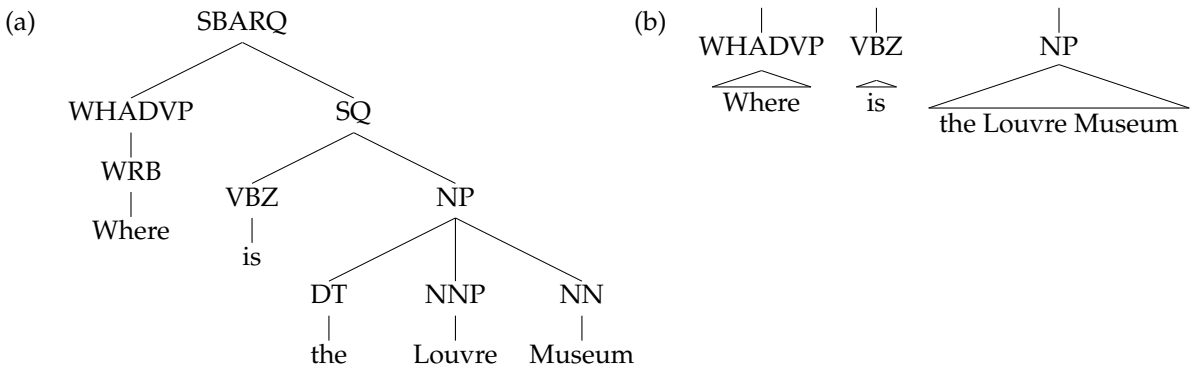

Figure 4.2: (a) Parse tree of the question *Where is the Louvre Museum ?*. (b) Phrasal nodes of the parse tree in (a).

pattern is then validated, by submitting the query "*the CN Tower is located in Toronto*" to Google search and dividing the number of results by the expected number of results. Table 4.1 presents some of the final patterns for this example.

Table 4.1: Example of extracted question rewrite patterns with respective precision.

Precision	Pattern
1.0	*NP* in {*ANSWER*} *VBZ*
0.75	*NP VBZ* located in {*ANSWER*}
1.0	*NP VBZ* one of {*ANSWER*}
1.0	{*ANSWER*} *VBZ NP*

4.4 Query submission

Once the information source to be used has been identified, and the queries have been formulated, the final step in the passage retrieval module is to submit the queries to the information source's end-point and retrieve the results. These queries are all submitted and processed in parallel, using multiple threads, in order to maximize the performance of the system.

Finally, after every query has finished, the results of each are aggregated (with duplicates removed), and sent to the answer extraction module for further processing. Metadata about the results, such as the rank of each search result, is also stored.

4.5 Conclusions

This chapter presented the passage retrieval component of the `QAML` system, which comprises two stages: query formulation and query submission. In the former, we developed a diverse set of strategies to transform a question into a suitable representation that can be sent to an information source, in order

to retrieve relevant passages for use in the answer extraction component. In particular, we showed how to leverage the use of DBpedia and Wikipedia to extract relevant information for non-factoid questions and, moreover, we also proposed a novel approach for bootstrapping lexico-syntactic query rewrite patterns. Although very few patterns were learnt and used in the final system, informal experiments led us to conclude that these patterns are indeed helpful for question answering. As for the latter stage – query submission –, queries are submitted in parallel using multiple threads, to maximize the performance of the system, as this stage is usually the costlier in the entire QA pipeline.

5 Answer extraction

This chapter presents the last component of the QAML system – answer extraction –, plus an evaluation of the overall performance of the system. Section 5.1 takes a second look at the answer extraction task, and discusses this work's approach to it. Section 5.2 examines the strategies that were employed to extract candidate answers from passages. Section 5.3 discusses how a final answer is selected from a set of candidate answers. Lastly, in Section 5.4, conclusions are drawn.

5.1 Introduction

Once a question has been classified, and relevant passages have been retrieved, the last step in a question answering system is to extract candidate answers from the passages, and select a final answer. We refer to this step as *answer extraction*. As an example, consider the question "*When was Mozart born ?*", classified as NUMERIC:DATE, and for which the following passages were retrieved:

- *Mozart Wolfgang Amadeus Mozart (born in Salzburg, Austria, on* ***January 27, 1756*** *– died in Vienna, Austria, on* ***December 5, 1791****) was a great composer.*
- *Mozart was born* ***January 27, 1756*** *in Salzburg, Austria.*

Given the above information, the answer extraction component would be responsible for extracting candidate answers from the passages – e.g., the instances of dates in bold face –, and then pick out the correct answer, which is *January 27, 1756*.

5.1.1 Our approach

Conceptually, the answer extraction process can be divided into two stages – candidate answer extraction and answer selection. The strategies adopted in this work for each stage are described in Section 5.2 and Section 5.3, respectively.

Briefly, for candidate answer extraction, we take advantage of the rich question type taxonomy (see Chapter 3) utilised in this work, to devise strategies for each particular 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.

As for answer selection, our strategy is to first filter out unwanted candidate answers, and then apply a clustering algorithm to group together similar answers. Finally, each resulting cluster is scored, and the final answer is decided by *ballot* – i.e., the answer within the cluster with highest score is chosen.

5.2 Candidate answer extraction

In the subsections that follow, we describe the strategies that were developed to extract plausible answers from relevant passages, as well as the question categories that each strategy is able to cope with.

5.2.1 Regular expressions

A regular expression is a sequence of symbols that can be used to specify text search strings. For example, to search for a sequence of one or more digits, the regular expression `/[0-9]+/` can be used. The reader is referred to (Jurafsky & Martin, 2008) for a detailed discussion on regular expressions.

We developed an extensive set of regular expressions to extract candidate answers for NUMERIC type questions, as they 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 [1] be utilised.

The 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, in this example. Each expression is also paired with a numeric score, which is assigned to every candidate answer that is matched and extracted by it.

5.2.2 Named entity recognizer

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 question categories present in this work's question type taxonomy require particular names as candidate answers – e.g., HUMAN:INDIVIDUAL questions call for person names –, which can occur in many different formats,

[1] It is worth mentioning that the regular expressions utilised in this work are far more complex than the expressions presented in the examples we have provided so far, and take in consideration a wide range of numeric formats and units.

Named entity type	Question category
PERSON	HUMAN:INDIVIDUAL
LOCATION	LOCATION:CITY LOCATION:COUNTRY LOCATION:MOUNTAIN LOCATION:OTHER LOCATION:STATE
ORGANIZATION	HUMAN:GROUP
MISCELLANEOUS	ENTITY:CREATIVE ENTITY:EVENT

Table 5.1: Mappings between named entity types and question categories.

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.

Table 5.1 presents the mappings between named entity types and question categories that we have adopted in this work. For each question category, we consider all extracted entities of the mapped type as candidate answers.

5.2.3 WordNet-based recognizer

So far, we have considered numerical answers – for which regular expressions are a good fit –, and entity-based answers – dealt with by a machine learning-based named entity recognizer. There is, however, another type of answer that we consider in this work, and which does not fall in either of the above categories – *type of* questions. Consider the ENTITY:ANIMAL question "*Which **animal** is the fastest ?*". For 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, as they can be easily confused with other nouns that are present in relevant passages.

In this work, we suggest a new approach for extracting answers for *type of* questions, using WordNet's hyponymy relations. We exploit the fact that candidate answers for these questions are often

animal, animate being, beast, brute, creature, fauna
⇐ domestic animal
  ⇐ house cat
  ⇐ dog
⇐ predatory animal
  ⇐ carnivore
    ⇐ feline
      ⇐ leopard
      ⇐ jaguar
      ⇐ **cheetah**
      ⇐ lion

Figure 5.1: Sample hyponym tree for the synset *animal*.

hyponyms of the question's headword, as illustrated from the above example and Figure 5.1, to construct an *online* [2] dictionary 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 utilised for every question that pertains to the ENTITY category, with the exception of ENTITY:EVENT, ENTITY:LETTER, ENTITY:TERM, and ENTITY:WORD. Moreover, it is also used for the categories HUMAN:TITLE, LOCATION:MOUNTAIN, and LOCAITON:OTHER.

5.2.4 Gazetteer

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 [3] 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 utilised in a similar way as the online exact dictionary matcher described in the previous section, with the difference being in the fact that gazetteers are not constructed online, but rather at the system's startup.

[2] Within this context, online refers to the fact that the dictionaries are not constructed *a priori*, but rather when they are needed.
[3] A gazetteer is a geographical dictionary, tipically used to identify places. However, we use the term in a more broader sense to refer to a dictionary of any type of entities.

5.3 Answer selection

After candidate answers have been extracted, the last step is to pick out a final answer to be returned. This is accomplished through three steps: candidate answer filtering, candidate answer clustering, and scoring. Each of these steps is described in the following three sections.

5.3.1 Candidate answer filtering

In order to remove undesired candidate answers, we apply a simple filter which consists of removing candidate answers which are contained in the original question. To understand the importance of this filter, consider the question "*Who is Tom Cruise's wife ?*", classified as HUMAN:INDIVIDUAL. For this question, *Tom Cruise* is extracted as a candidate answer, since it matches the named entity type associated with the question's category – PERSON –, even though it is clearly not the answer that is sought. Moreover, this answer appears in almost every passage, as the formulated query itself contains *Tom Cruise*, which will result in a very high score for it. Thus, in order to prevent this unwanted answer to be returned, a filter that removes any candidate answer that is contained in the original question is required.

5.3.2 Candidate answer clustering

Once unwanted candidate answers have been filtered, we perform clustering on the remaining candidates. Clustering is an example of unsupervised learning, whose goal is to group similar instances together, into a set of clusters. But before delving any further, one needs to define a distance measure, which determines how the similarity of two candidate answers is calculated. In this work, we use the *overlap distance*, defined as:

$$\text{overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}\,, \tag{5.1}$$

where $|X \cap Y|$ is the number of tokens shared by candidate answers X and Y, and $\min(|X|, |Y|)$ is the size of the smallest candidate answer being compared. This metric 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).

The overlap 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 are merged up to a specified threshold distance, which we defined to be 0.0. The distance 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, consider the following set of candidate answers: {John Kennedy, Kennedy, John F. Kennedy, John McCarthy}. In the first step, *John Kennedy* and *Kennedy* are merged together. In the second step, *John F. Kennedy* is merged with the resulting cluster from the previous step. Finally, since the minimum distance from *John McCarthy* to any member in the cluster {John Kennedy, Kennedy, John F. Kennedy} is 0.5, and this value is greater than the threshold, the algorithm halts.

In addition, clustering is also used to select the most informative answer, by returning the longest answer within each cluster. For instance, in the cluster {John Kennedy, Kennedy, Joyn F. Kennedy}, *John F. Kennedy* would be selected if this cluster happened to be chosen as the answer.

5.3.3 Scoring

In the final scoring step, a score is assigned to each cluster, which is simply the sum of the scores [4] of all candidate answers within it. Then, the longest answer within 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 cluster with highest search rank – i.e., the cluster whose answers were in the first results returned by the information source.

5.4 Conclusions

In this chapter, we presented a two-phased approach to answer extraction. For the first stage, candidate answer extraction, we utilised several extraction techniques, ranging from handcrafted regular expressions to automatic machine learning-based named entity recognizers. We have also presented a novel approach to extract candidate answers, which makes use of WordNet's hyponymy relations to construct an online exact dictionary matcher, allowing our system to cover a wide range of question categories. As for the second stage, answer selection, we start by removing unwanted candidate answers, which are then clustered together using an agglomerative clustering algorithm, and finally scored to select an answer.

[4] With the exception of candidate answers that were extracted using regular expressions, every other candidate answer has a score of 1.0, so the score of each cluster boils down to the number of members within it, in most scenarios.

6 Evaluation

Now that we have presented all the components that comprise the QAML system, we turn our attention to its evaluation. In Section 6.1, we describe the experimental setup that was used for performance evaluation, namely, the question data set and the evaluation measures. Then, in Section 6.2, we report and discuss results. Finally, in Section 6.3, we draw conclusions.

6.1 Experimental setup

6.1.1 Data set

To evaluate the system, we collected a test set of 50 questions from potential users of a question answering system. These questions are listed in Table 6.1.

6.1.2 Evaluation measures

To assess the performance of the system, we used mean reciprocal rank (MRR) as the evaluation measure, as described in Chapter 2. For a test set of N questions, MRR is defined as:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i}, \tag{6.1}$$

where $rank_i$ is the rank of the returned answer for the i-th question in the test set. Thus, in order to use MRR, our system is required to return a ranked list of possible answers for a given question. Also, if the system is unable to answer a given question, the reciprocal rank is deemed as zero.

6.2 Experimental results

We evaluated the question answering system using two different settings: (1) considering only the first returned answer; and (2) considering the first three answers returned by the system. The results for each question in the test set are shown in Table 6.1.

Table 6.1: List of test questions that were used for the evaluation of the system, plus the rank of the correct answer (zero represents an unanswered question).

Question	Answer rank
What famous statue was given by France to the USA?	1
What language do they speak in Rome?	1
When was the BIC pen invented?	0
How high is Kilimanjaro?	1
When was Salazar Bridge renamed to 25th of April Bridge?	1
What is Lance Armstrong known for doing?	0
Who is referred to as The artist formerly known as Prince?	0
What is the nationality of Ray Charles?	1
Who features Madonna in “4 Seconds to Save the World”?	1
Which will be the host city of the Olympic Games in 2016?	1
Which is the most spoken romance language?	1
Where can I eat Fish and Chips?	2
How many times a day should a Muslim pray?	1
Who was recently laureated with the Nobel Peace Prize?	1
How many legs has a turtle?	1
How many colors has the rainbow?	1
Who was the founder of IKEA?	1
How many letters are there in the Hungarian alphabet?	1
How far is it from Jupiter to Saturn?	3
Who wrote Alice’s Adventures in Wonderland?	1
Who married Gwen Stefani?	1
How many times did Elizabeth Taylor divorce?	0
What is the color of Rudolph Reindeer’s nose?	1
What is the best selling album of all time?	0
Who invented the dynamo?	1
When was Vasco da Gama born?	1
What is the fruit harvested from apple tree?	0
How many days has a year?	2
When was the first linux distribution made?	0
What was the nationality of Ferenc Puskás?	1
In which year died Freddy Mercury?	1
How many centimetres are one inch?	0
When do we celebrate new year’s eve?	0
Which is the fastest animal?	1
Who painted Mona Lisa?	1
Who wrote Gone with the wind?	1
Who wrote the ”Lord of the rings”?	1
What is the name of the portuguese prime-minister?	1
Who killed Darth Vader?	0
Who is the writer of Noddy books?	1
What is the last season of Lost?	0
How old was Socrates when he died?	1
Who is the tallest man in the world?	1
What is the name of Harry Potter’s mother?	0
Who wrote Harry Potter?	1
How many wives did Henry VIII kill?	1
What is the capital of Lithuania?	1
How tall is Big Ben?	1
How old is Manuel de Oliveira?	1
What is Manuel de Oliveira’s age?	0

For the first setting, we attained a mean reciprocal rank of $\mathbf{0.68}$, which means that our system correctly answered 34 out of the 50 questions of the test set. As for the second setting, that is, when considering answers that were found in the top three results returned by the system, we obtained a mean reciprocal rank of $\mathbf{0.70(6)}$, and the system was able to correctly answer 37 questions.

6.2.1 Discussion

We now analyze the experimental results from different perspectives, and discuss how they can be improved.

6.2.1.1 Answer ranks

First, let us consider questions for which a correct answer was found, but was not ranked first, such as "*Where can I eat Fish and Chips?*". For this question, the ranked list of returned answers was {British, **London**, Chicago}, extracted from the following (ordered by information source relevance) passages:

- *Traditionally,* ***British*** *fish and chip shops offer free wooden forks to customers so that they can eat their chips.*
- *In 1860 The first fish and chip shop was opened in* ***London*** *by Jewish.*
- *Come to NBC* ***Chicago*** *for $8.95 All-You-Can-Eat Fish & Chips.* [1]

The returned answers all have a score of $\mathbf{1.0}$, but since our tie-breaker algorithm takes in consideration the relevance of the passages where the answer was found, *British* is ranked first. The astute reader might argue that *British* is not a location and therefore should not be a candidate answer for LOCATION:OTHER questions, such as this one. However, machine learning named entity recognizers, such as the one employed in this work, typically identify nationalities (demonyms) as locations. A possible improvement would be to retrain the model, using a modified annotated corpus that does not have demonyms labeled as locations.

As for the question "*How far is it from Jupiter to Saturn?*", the reasons for not returning the correct answer in the first result differ from the above. For this question, the problem was due to the fact that the first retrieved passages had the distance from both planets to the Sun, instead of the distance between both planets. To solve this problem, the scoring function could be further improved, by taking into consideration additional information, such as the distance from the query keywords to the candidate answer.

[1] *Chicago* is actually a valid answer, but it was not the one that was expected by the user that posed the question.

6.2.1.2 Unanswered questions

We now analyze some questions for which our system was unable to return the correct answer. Consider, for example, the question "*What is Manoel de Oliveira's age ?*". In this case, the system identifies some passages that do contain Manoel de Oliveira's age, but that were written three years ago, which results in an incorrect age of 97 instead of 100. For this particular type of questions, a possible improvement would be to rephrase it as "*When was Manoel de Oliveira born?*" – whose answer is easily extracted – , and then compute his current age from the extracted date. A similar reasoning can also be applied to the question "*How old is Manoel de Oliveira?*", even though our system was able to answer it. This is because the answer that was extracted – 100 –, will potentially be invalid next year, as Manoel de Oliveira celebrates his **101**-th birthday.

For other unanswered questions, such as "*When was the first Linux distribution made?*", a better query formulation strategy is required, since the correct answer could not be found in the first results returned by Google search, and thus, our system was unable to extract it.

6.2.1.3 Practical impact of the proposed techniques

The techniques proposed by this work also proved to be of great importance, as many questions in the test set were correctly answered by making use of them. For instance, the WordNet-based recognizer was able to extract dictionaries that held the answer to questions such "*Which is the **fastest animal**?*", "*What famous **statue** was given by France to the USA?*", "*What **language** do they speak in Rome?*", *et cetera*.

Regarding candidate answer filtering, the technique was useful for questions such as "*Who married Gwen Stefani?*" and "*Who features Madonna in "4 Seconds to Save the World"?*", for which extracted candidate answers are present in the question. Without this technique, *Gwen Stefani* and *Madonna* would be incorrectly extracted as answers to the above questions. However, filtering also prevented the system from answering "*What is the fruit harvested from apple tree?*", because the correct answer – *apple* – was contained in the question. Nevertheless, the benefits of this technique far outweigh the disadvantages.

Candidate answer clustering have also proved its usefulness, by allowing the system to distinguish *facts from fiction*. For instance, for the question "*Who wrote Harry Potter?*", clustering allowed us to distinguish the real writer – *J. K. Rowling* –, from an actor – *Daniel Radcliffe* –, that played the part of *Harry Potter* in a movie. Since the real writer appears more often in the passages – sometimes with variations such as *Rowling* and *J. K. Rowling* –, the single cluster containing all variations is returned as the first answer. Filtering was also applied in this question to remove *Harry Potter* from the candidate answers.

The downside of clustering is that, sometimes, it is overly *aggressive*, in the sense that it clusters candidate answers that it was not supposed to. For instance, consider the question "*Who killed Darth*

Vader?", and the following three candidate answers {Luke Skywalker, Skywalker, Anakin Skywalker}. Although *Anakin Skywalker* [2] and *Luke Skywalker* refer to different entities, they are both assigned to the same cluster by way of *Skywalker*. Then, since *Anakin Skywalker* is the longest answer within the cluster, it is wrongly returned as the answer, instead of *Luke Skywalker*. To try to cope with this problem, a complete-link strategy could be used in the clustering algorithm, or a different – less *aggressive* – distance function.

6.2.1.4 Web Zeitgeist

Another interesting observation, which is a by-product of using information from the Web to answer questions, is that the answer the system returns for a given question may vary from time to time, as new data is (re)indexed by the information sources. For instance, for the question "*Who is the tallest man in the world?*", the first time our system was evaluated, the answer *Bao Xishun* was returned, who was once considered the tallest man in the world. However, for the second evaluation, which was carried just a few days apart from the first, the correct answer *Turk Sultan Kosen* was returned, as news start spreading that he was going to be appointed by the Guiness Book as the tallest man in the world. The question "*Who was recently laureated with the Peace Nobel Prize?*" also shares similar traits with the above, since the second evaluation was carried out just a few days after a new person – Barack Obama – has been awarded with the prize.

6.3 Conclusions

In this chapter, we presented the results of the empirical evaluation of the `QAML` system. We concluded that the proposed techniques are indeed helpful to accurately extract answers, especially the WordNet-based recognizer and the clustering technique which allows the system to separate *facts from fiction*, by boosting the score of the most frequently occurring answers. Nevertheless, we have also identified some flaws in our system, for which we have suggested possible improvements.

Furthermore, it is also worth mentioning that all test questions were correctly classified (with the exception of "*What is the last season of Lost?*"), which proved to be very helpful in narrowing down the number of possible candidate answers that the answer extraction component needed to consider.

[2] For the curious reader, Anakin Skywalker *is* Darth Vader.

7 Conclusions and Future work

In this chapter, we conclude this thesis by outlining our main contributions to the question answering field in Section 7.1, by discussing directions for future work in Section 7.2, and by presenting our final remarks in Section 7.3.

7.1 Contributions to question answering

The main contributions of the present thesis to the Question Answering field are summarized in the following subsections.

7.1.1 Question classification

Contribution 1 A state-of-the-art, machine learning-based question classifier, that reaches an accuracy of 95.2% and 90.6% for coarse- and fine-grained classification, respectively, using the standard training and test corpus of Li&Roth. Furthermore, we have also performed thorough experiments to fine-tune the classifier, using several different combinations of lexical, syntactic, and semantic features. Experimental results led us to conclude that the most discriminative set of features for question classification are unigrams, question headword, and semantic headword, with a special focus on the latter two, for which several algorithms were developed to accurately extract them from a question.

7.1.2 Query formulation

Contribution 2 A novel approach for bootstrapping lexico-syntactic query rewrite patterns, based on the works of (Ravichandran & Hovy, 2001) and (Brill et al., 2002). In particular, the developed algorithm is able to learn how to rewrite a question, in a way that allows the system to retrieve passages which are very likely to contain an answer, thus easing the answer extraction task. To do so, our algorithm requires only two seed question-answer pairs as input, as well as a natural language parser.

Contribution 3 We showed how to leverage the combination of DBpedia and Wikipedia to extract answers for non-factoid questions, such as DESCRIPTION:DEFINITION and HUMAN:DESCRIPTION.

7.1.3 Answer extraction

Contribution 4 A wide range of strategies to extract candidate answers from relevant passages, ranging from handcrafted regular expressions to automatic machine learning-based named entity recognizers. In particular, we developed a novel approach to extract candidate answers, which makes use of WordNet's hyponymy relations to construct an online exact dictionary matcher, which allows the system to support a wide range of question categories.

Contribution 5 A three-step answer selection algorithm, which first filters irrelevant candidate answers, then clusters together the remaining, so that similar candidate answers are considered as one (with a combined score), and finally selects the candidate answer with highest score, using a simple heuristic to settle ties.

7.2 Future work

Throughout the preceding four chapters, we have already suggested potential improvements that could be made to the system developed in this work. In this section, we present possible improvements, which we consider most relevant for future research endeavors.

Extended question type taxonomy Chapter 3 stressed the importance of a diverse question type taxonomy in filtering irrelevant answer candidates and, although the taxonomy employed in this work is considerably broad in scope, it could be extended to provide even more fine control of the extracted answers' answer type. For example, one might argue why is there a specific LOCATION:MOUNTAIN category, but not LOCATION:RIVER. On the other side, there are also some categories which are too generic, such as NUMERIC:DATE. Using this category solely, our system is unable to distinguish between questions such as "*What day was Michael Jackson born on?*", which asks for a specific day, and "*In what year was Michael Jackson born?*" which asks for a particular year. Thus, we propose a thorough revision of the question type taxonomy in future works.

Anaphora resolution In the 2004 Question Answering track of the TREC competition (Voorhees, 2004), questions were grouped into different series, where each series was associated with a topic. For example, one of the topics was *Hale Bopp comet* and some of the questions about it were "*When was **the comet** discovered?*" and "*How often does **it** approach the earth?*". The words in bold face are examples of anaphoric expressions, that is, expressions that refer to another – *Hale Bopp comet* in this particular case. Therefore, in order to understand what is being asked, it is necessary to find out what the anaphora is referring to. Currently, our system doesn't support these questions. A further improvement of the developed system would be to include anaphora resolution techniques (Jurafsky & Martin, 2008).

Support for non-factoid questions Currently, for fine-grained categories under DESCRIPTION, our system only supports DESCRIPTION:DEFINITION and HUMAN:DESCRIPTION questions which, as we have seen, are answered using a strategy that makes use of Wikipedia and DBpedia. Other non-factoid questions, however, such as DESCRIPTION:MANNER an DESCRIPTION:REASON, are not currently supported, for they require a very advanced kind of reasoning, which is beyond the scope of this thesis. For instance, for the question "*How do you get to the top of the Eiffel tower ?*", the system would have to extract several steps, and then combine them into a structured plan which would be returned as the answer. It would be interesting to explore techniques that deal with these particular questions, and incorporate them in our system.

Extended system evaluation In this work, we evaluated both the question classification component in isolation, and the overall performance of the system. For the latter, although we have obtained good results with real questions from potential users of the system, we believe the evaluation could be extended, in order to understand what components need most improvement, and to determine the individual impact of each particular technique. A similar evaluation was performed by (Moldovan et al., 2003). For instance, it would to be interesting to evaluate the performance of the system without and with clustering, to determine the impact of this technique in the system. Moreover, for future evaluations, we also suggest the use of a larger and more diverse question test set, so that every question category in the question type taxonomy can be evaluated.

7.3 Conclusions

In this thesis, we have presented a multi-faceted approach to web question answering, with a special focus on machine learning techniques. We have proposed several techniques for each component of the developed system, some of which are novel in the question answering field. These techniques, summarized in Section 7.1, have also proved to be very effective for question answering.

From the results of this work, we conclude that is indeed possible to successfully apply machine learning techniques in every component of a question answering system. However, it is also worth mentioning that machine learning techniques are not enough by themselves and, in order for them to be really effective and to achieve over-the-top results, they need to be complemented with natural language processing techniques. For instance, in the question classification task, we have shown that a machine learning-based classifier using solely superficial features is unable to surpass the accuracy *barrier* of 90.0%. However, when complementing these features with more advanced ones – such as the question headword –, that require the use of natural language processing techniques, we are able to achieve a state-of-the-art accuracy of 95.2%. This answers the first research question of this work.

As for the second research question, we have proposed several different techniques that leverage

the vast amount of information available in the Web to accurately answer natural language questions. First, the bootstrapping technique, which only works because of the amount of redundancy available in the Web. Second, the use of different search strategies, depending on the question category, which is only possible due to the many information sources available in the Web. Finally, almost every technique used in the answer extraction layer implicitly takes advantage of the vast amount of information in the Web, by *blindly* extracting answers from passages, and assuming that the answer that appears more often is the correct one.

I Appendix

A Resources

A.1 WordNet mappings

```
<?xml version="1.0" encoding="UTF-8"?>
<map wn-version="3.0" xmlns:xsi="http://www.w3.org/2001/XMLSchema-instance" xsi:
    noNamespaceSchemaLocation="WordNetMapSchema.xsd">
  <!-- ABBREVIATION -->
  <category name="ABBREVIATION_ABBREVIATION">
    <target offset="7091587" POS="noun" words="abbreviation -- (a shortened form of a word or
        phrase)"/>
    <target offset="7091902" POS="noun" words="acronym -- (a word formed from the initial
        letters of the several words in the name)"/>
  </category>
  <category name="ABBREVIATION_EXPANSION" />
  <!-- DESCRIPTION -->
  <category name="DESCRIPTION_DEFINITION">
    <target offset="6601327" POS="noun" words="meaning, significance, signification, import --
        (the message that is intended or expressed or signified)" />
    <target offset="5919866" POS="noun" words="meaning, substance -- (the idea that is
        intended)" />
    <target offset="6744396" POS="noun" words="definition -- (a concise explanation of the
        meaning of a word or phrase or symbol)" />
    <target offset="957378" POS="verb" words="define -- (give a definition for the meaning of
        a word)" />
  </category>
  <category name="DESCRIPTION_DESCRIPTION">
    <target offset="13559177" POS="noun" words="origin -- (the source of something's existence
        or from which it derives or is derived)" />
    <target offset="4748836" POS="noun" words="difference -- (the quality of being unlike or
        dissimilar)" />
    <target offset="987071" POS="noun" words="describe, depict, draw -- (give a description of
        )" />
  </category>
  <category name="DESCRIPTION_MANNER" />
  <category name="DESCRIPTION_REASON">
    <target offset="9178999" POS="noun" words="reason, ground -- (a rational motive for a
        belief or action)" />
  </category>
```

```
<!— ENTITY —>
<category name="ENTITY_ANIMAL">
  <target offset="15388" POS="noun" words="animal, animate_being, beast, brute, creature,
      fauna — (a living organism characterized by voluntary movement)" />
  <target offset="7993929" POS="noun" words="animal_group — (a group of animals)"/>
</category>
<category name="ENTITY_BODY">
  <target offset="5220461" POS="noun" words="body_part — (any part of an organism such as
      an organ or extremity)" />
</category>
<category name="ENTITY_COLOR">
  <target offset="4956594" POS="noun" words="color, colour, coloring, colouring — (a visual
       attribute of things that results from the light they emit or transmit or reflect)" />
</category>
<category name="ENTITY_CREATIVE">
  <target offset="6619065" POS="noun" words="show — (a social event involving a public
      performance or entertainment)" />
  <target offset="7020895" POS="noun" words="music — (an artistic form of auditory
      communication incorporating instrumental or vocal tones in a structured and continuous
       manner)" />
  <target offset="6410904" POS="noun" words="book — (a written work or composition that has
       been published (printed on pages bound together))" />
  <target offset="6362953" POS="noun" words="writing, written_material, piece_of_writing —
      (the work of a writer; anything expressed in letters of the alphabet (especially when
      considered from the point of view of style and effect))" />
  <target offset="3038595" POS="noun" words="classic — (a creation of the highest
      excellence)" />
  <target offset="2743547" POS="noun" words="art, fine_art — (the products of human
      creativity; works of art collectively)" />
  <target offset="6254669" POS="noun" words="medium — (a means or instrumentality for
      storing or communicating information)" />
</category>
<category name="ENTITY_CURRENCY">
  <target offset="13385913" POS="noun" words="currency — (the metal or paper medium of
      exchange that is presently used)" />
</category>
<category name="ENTITY_MEDICINE">
  <target offset="14299637" POS="noun" words="symptom — ((medicine) any sensation or change
       in bodily function that is experienced by a patient and is associated with a
      particular disease)" />
  <target offset="3247620" POS="noun" words="drug — (a substance that is used as a medicine
       or narcotic)" />
  <target offset="657604" POS="noun" words="medical_care, medical_aid — (professional
      treatment for illness or injury)" />
  <target offset="14034177" POS="noun" words="physical_condition, physiological_state,
      physiological_condition — (the condition or state of the body or bodily functions)" /
```

```
      >
</category>
<category name="ENTITY_EVENT">
  <target offset="29378" POS="noun" words="event -- (something that happens at a given place
       and time)" />
</category>
<category name="ENTITY_FOOD">
  <target offset="21265" POS="noun" words="food, nutrient -- (any substance that can be
      metabolized by an animal to give energy and build tissue)" />
  <target offset="7555863" POS="noun" words="food, solid_food -- (any solid substance (as
      opposed to liquid) that is used as a source of nourishment)" />
  <target offset="13134947" POS="noun" words="fruit -- (the ripened reproductive body of a
      seed plant)"/>
  <target offset="7578363" POS="noun" words="helping, portion, serving -- (an individual
      quantity of food or drink taken as part of a meal)" />
</category>
<category name="ENTITY_INSTRUMENT">
  <target offset="3800933" POS="noun" words="musical_instrument, instrument -- (any of
      various devices or contrivances that can be used to produce musical tones or sounds)"
      />
</category>
<category name="ENTITY_LANGUAGE">
  <target offset="6282651" POS="noun" words="language, linguistic_communication -- (a
      systematic means of communicating by the use of sounds or conventional symbols)"/>
  <target offset="7155661" POS="noun" words="dialect, idiom, accent -- (the usage or
      vocabulary that is characteristic of a specific group of people)"/>
</category>
<category name="ENTITY_LETTER">
  <target offset="6828818" POS="noun" words="letter, letter_of_the_alphabet,
      alphabetic_character -- (the conventional characters of the alphabet used to represent
       speech)"/>
  <target offset="6488880" POS="noun" words="character_set -- (an ordered list of characters
       that are used together in writing or printing)"/>
</category>
<category name="ENTITY_PLANT">
  <target offset="17222" POS="noun" words="plant, flora, plant_life -- ((botany) a living
      organism lacking the power of locomotion)"/>
  <target offset="8436759" POS="noun" words="vegetation, flora, botany -- (all the plant
      life in a particular region or period)"/>
</category>
<category name="ENTITY_PRODUCT">
  <target offset="6845599" POS="noun" words="trade_name, brand_name, brand, marque -- (a
      name given to a product or service)" />
  <target offset="3748886" POS="noun" words="merchandise, ware, product -- (commodities
      offered for sale)" />
</category>
```

```
<category name="ENTITY_RELIGION">
  <target offset="5946687" POS="noun" words="religion, faith, religious_belief -- (a strong
      belief in a supernatural power or powers that control human destiny)"/>
</category>
<category name="ENTITY_SPORT">
  <target offset="523513" POS="noun" words="sport, athletics -- (an active diversion
      requiring physical exertion and competition)"/>
  <target offset="455599" POS="noun" words="game -- (a contest with rules to determine a
      winner)"/>
  <target offset="430606" POS="noun" words="game -- (an amusement or pastime)"/>
  <target offset="624738" POS="noun" words="exercise, exercising, physical_exercise,
      physical_exertion, workout -- (the activity of exerting your muscles in various ways
      to keep fit)"/>
</category>
<category name="ENTITY_SUBSTANCE">
  <target offset="19613" POS="noun" words="substance -- (the real physical matter of which a
       person or thing consists)"/>
</category>
<category name="ENTITY_SYMBOL">
  <target offset="6806469" POS="noun" words="symbol -- (an arbitrary sign (written or
      printed) that has acquired a conventional significance)" />
</category>
<category name="ENTITY_TECHNIQUE">
  <target offset="5665146" POS="noun" words="technique -- (a practical method or art applied
       to some particular task)" />
  <target offset="941140" POS="noun" words="approach, attack, plan_of_attack -- (ideas or
      actions intended to deal with a problem or situation)" />
  <target offset="4928903" POS="noun" words="manner, mode, style, way, fashion -- (how
      something is done or how it happens)" />
  <target offset="5660268" POS="noun" words="method -- (a way of doing something, especially
       a systematic way)" />
  <target offset="1023820" POS="noun" words="procedure, process -- (a particular course of
      action intended to achieve a result)" />
</category>
<category name="ENTITY_TERM">
  <target offset="6303888" POS="noun" words="term -- (a word or expression used for some
      particular thing)" />
</category>
<category name="ENTITY_VEHICLE">
  <target offset="4524313" POS="noun" words="vehicle -- (a conveyance that transports people
       or objects)" />
</category>
<category name="ENTITY_WORD">
  <target offset="6286395" POS="noun" words="word -- (a unit of language that native
      speakers can identify)" />
</category>
```

```
<category name="ENTITY_OTHER" />
<!-- HUMAN -->
<category name="HUMAN_DESCRIPTION" />
<category name="HUMAN_GROUP">
  <target offset="8008335" POS="noun" words="organization, organisation -- (a group of
      people who work together)" />
  <target offset="31264" POS="noun" words="group, grouping -- (any number of entities (
      members) considered as a unit)" />
</category>
<category name="HUMAN_INDIVIDUAL">
  <target offset="7846" POS="noun" words="person, individual, someone, somebody, mortal,
      soul -- (a human being)" />
  <target offset="2472293" POS="noun" words="homo, man, human_being, human -- (any living or
       extinct member of the family Hominidae characterized by superior intelligence,
      articulate speech, and erect carriage)"/>
  <target offset="6333653" POS="noun" words="name -- (a language unit by which a person or
      thing is known)" />
  <target offset="9504135" POS="noun" words="spiritual_being, supernatural_being -- (an
      incorporeal being believed to have powers to affect the course of human events)" />
  <target offset="10636598" POS="noun" words="spirit -- (the vital principle or animating
      force within living things)"/>
</category>
<category name="HUMAN_TITLE">
  <target offset="8112096" POS="noun" words="profession -- (the body of people in a learned
      occupation)" />
  <target offset="6339416" POS="noun" words="title, title_of_respect, form_of_address -- (an
       identifying appellation signifying status or function: e.g. 'Mr.' or 'General')" />
  <target offset="582388" POS="noun" words="occupation, business, job, line_of_work, line --
       (the principal activity in your life that you do to earn money)" />
</category>
<!-- LOCATION -->
<category name="LOCATION_CITY">
  <target offset="8675967" POS="noun" words="urban_area, populated_area -- (a geographical
      area constituting a city or town)"/>
</category>
<category name="LOCATION_COUNTRY">
  <target offset="8168978" POS="noun" words="state, nation, country, land, commonwealth,
      res_publica, body_politic -- (a politically organized body of people under a single
      government)"/>
  <target offset="8544813" POS="noun" words="country, state, land -- (the territory occupied
       by a nation)"/>
</category>
<category name="LOCATION_MOUNTAIN">
  <target offset="9359803" POS="noun" words="mountain, mount -- (a land mass that projects
      well above its surroundings)"/>
  <target offset="9470550" POS="noun" words="vent, volcano -- (a fissure in the earth's
```

```
      crust (or in the surface of some other planet) through which molten lava and gases
      erupt)"/>
  <target offset="9403734" POS="noun" words="range, mountain_range, range_of_mountains,
      chain, mountain_chain, chain_of_mountains -- (a series of hills or mountains)"/>
</category>
<category name="LOCATION_STATE">
  <target offset="8654360" POS="noun" words="state, province -- (the territory occupied by
      one of the constituent administrative districts of a nation)"/>
</category>
<category name="LOCATION_OTHER">
  <target offset="27167" POS="noun" words="location -- (a point or extent in space)"/>
  <target offset="9334396" POS="noun" words="land, dry_land, earth, ground, solid_ground,
      terra_firma -- (the solid part of the earth's surface)"/>
  <target offset="9225146" POS="noun" words="body_of_water, water -- (the part of the earth'
      s surface covered with water (such as a river or lake or ocean)"/>
  <target offset="9239740" POS="noun" words="celestial_body, heavenly_body -- (natural
      objects visible in the sky)"/>
</category>
<!-- NUMERIC -->
<category name="NUMERIC_CODE">
  <target offset="6426111" POS="noun" words="phone_number, telephone_number, number -- (the
      number is used in calling a particular telephone)" />
  <target offset="6353934" POS="noun" words="code -- (a coding system used for transmitting
      messages requiring brevity or secrecy)" />
</category>
<category name="NUMERIC_COUNT">
  <target offset="13582013" POS="noun" words="number -- (a concept of quantity derived from
      zero and units)" />
</category>
<category name="NUMERIC_DATE">
  <target offset="15159583" POS="noun" words="date, day_of_the_month -- (the specified day
      of the month)" />
</category>
<category name="NUMERIC_DISTANCE">
  <target offset="5084201" POS="noun" words="distance -- (the property created by the space
      between two objects or points)" />
  <target offset="5002352" POS="noun" words="stature, height -- ((of a standing person) the
      distance from head to foot)" />
  <target offset="5093581" POS="noun" words="dimension -- (the magnitude of something in a
      particular direction (especially length or width or height))" />
  <target offset="13603305" POS="noun" words="linear_unit, linear_measure -- (a unit of
      measurement of length)" />
  <target offset="5129565" POS="noun" words="distance, length -- (size of the gap between
      two places)" />
  <target offset="5134547" POS="noun" words="depth, deepness -- (the extent downward or
      backward or inward)" />
```

```
</category>
<category name="NUMERIC_MONEY">
  <target offset="13275288" POS="noun" words="outgo, spending, expenditure, outlay -- (money
       paid out; an amount spent)" />
  <target offset="13384557" POS="noun" words="money -- (the most common medium of exchange;
      functions as legal tender)" />
  <target offset="5145118" POS="noun" words="monetary_value, price, cost -- (the property of
       having material worth (often indicated by the amount of money something would bring
      if sold))" />
  <target offset="13303315" POS="noun" words="price, terms, damage -- (the amount of money
      needed to purchase something)" />
  <target offset="13331198" POS="noun" words="sum, sum_of_money, amount, amount_of_money --
      (a quantity of money)" />
  <target offset="13604718" POS="noun" words="monetary_unit -- (a unit of money)" />
</category>
<category name="NUMERIC_ORDER" />
<category name="NUMERIC_OTHER">
  <target offset="33615" POS="noun" words="measure, quantity, amount -- (how much there is
      or how many there are of something that you can quantify)" />
</category>
<category name="NUMERIC_PERCENT">
  <target offset="13815152" POS="noun" words="magnitude_relation, quantitative_relation -- (
      a relation between magnitudes)" />
  <target offset="4756172" POS="noun" words="probability -- (the quality of being probable)"
       />
  <target offset="5091770" POS="noun" words="probability, chance -- (a measure of how likely
       it is that some event will occur)" />
</category>
<category name="NUMERIC_PERIOD">
  <target offset="15113229" POS="noun" words="time_period, period_of_time, period -- (an
      amount of time)" />
  <target offset="15154774" POS="noun" words="time_unit, unit_of_time -- (a unit for
      measuring time periods)" />
  <target offset="4924103" POS="noun" words="age -- (how long something has existed)" />
  <target offset="7309599" POS="noun" words="time, clip -- (an instance or single occasion
      for some event)" />
  <target offset="15142025" POS="noun" words="life_expectancy -- (an expected time to live
      as calculated on the basis of statistical probabilities)" />
</category>
<category name="NUMERIC_SPEED">
  <target offset="15282696" POS="noun" words="speed, velocity -- (distance travelled per
      unit time)" />
  <target offset="5058140" POS="noun" words="speed, swiftness, fastness -- (a rate (usually
      rapid) at which something happens)" />
</category>
<category name="NUMERIC_TEMPERATURE">
```

```
    <target offset="5011790" POS="noun" words="temperature -- (the degree of hotness or
        coldness of a body or environment (corresponding to its molecular activity))" />
    <target offset="13608598" POS="noun" words="temperature_unit -- (a unit of measurement for
         temperature)" />
  </category>
  <category name="NUMERIC_SIZE">
    <target offset="5098942" POS="noun" words="size -- (the physical magnitude of something (
        how big it is))" />
    <target offset="13779032" POS="noun" words="volume -- (the amount of 3-dimensional space
        occupied by an object)" />
    <target offset="5128519" POS="noun" words="area, expanse, surface_area -- (the extent of a
         2-dimensional surface enclosed within a boundary)" />
    <target offset="13600404" POS="noun" words="area_unit, square_measure -- (a system of
        units used to measure areas)" />
    <target offset="13600822" POS="noun" words="volume_unit, capacity_unit, capacity_measure,
        cubage_unit, cubic_measure, cubic_content_unit, displacement_unit, cubature_unit -- (a
         unit of measurement of volume or capacity)" />
  </category>
  <category name="NUMERIC_WEIGHT">
    <target offset="5026843" POS="noun" words="weight -- (the vertical force exerted by a mass
         as a result of gravity)" />
    <target offset="13609214" POS="noun" words="mass_unit -- (a unit of measurement for mass)"
         />
    <target offset="13608788" POS="noun" words="weight_unit, weight -- (a unit used to measure
         weight)" />
    <target offset="5024254" POS="noun" words="mass -- (the property of a body that causes it
        to have weight in a gravitational field)" />
    <target offset="2704818" POS="verb" words="weigh -- (have a certain weight)" />
  </category>
</map>
```

Bibliography

Alias-i. (2008). *LingPipe 3.8.1.* (Software available at `http://alias-i.com/lingpipe`)

Amaral, C., Cassan, A., Figueira, H., Martins, A., Mendes, A., Mendes, P., et al. (2008). Priberam's question answering system in qa@clef 2007. 364–371.

Androutsopoulos, I. (1995). Natural language interfaces to databases - an introduction. *Journal of Natural Language Engineering*, *1*, 29–81.

Auer, S., Bizer, C., Kobilarov, G., Lehmann, J., & Ives, Z. (2007). Dbpedia: A nucleus for a web of open data. In *In 6th int'l semantic web conference, busan, korea* (pp. 11–15). Springer.

Baeza-Yates, R. A., & Ribeiro-Neto, B. A. (1999). *Modern information retrieval*. ACM Press / Addison-Wesley.

Bert F. Green, J., Wolf, A. K., Chomsky, C., & Laughery, K. (1961). 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* (pp. 219–224). New York, NY, USA: ACM.

Bhagat, R., Leuski, A., & Hovy, E. (2005). *Shallow semantic parsing despite little training data.* Proceedings of the ACL/SIGPARSE 9th International Workshop on Parsing Technologies. Vancouver, B.C., Canada.

Blunsom, P., Kocik, K., & Curran, J. R. (2006). Question classification with log-linear models. In *Sigir '06: Proceedings of the 29th annual international acm sigir conference on research and development in information retrieval* (pp. 615–616). New York, NY, USA: ACM.

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Colt '92: Proceedings of the fifth annual workshop on computational learning theory* (pp. 144–152). New York, NY, USA: ACM.

Brill, E., Dumais, S., & Banko, M. (2002). An analysis of the askmsr question-answering system. In *Emnlp '02: Proceedings of the acl-02 conference on empirical methods in natural language processing* (pp. 257–264). Morristown, NJ, USA: Association for Computational Linguistics.

Brin, S., & Page, L. (1998). The anatomy of a large-scale hypertextual web search engine. In *Computer networks and isdn systems* (Vol. 30, pp. 107–117). Amsterdam, The Netherlands, The Netherlands: Elsevier Science Publishers B. V.

Carlson, A. J., Cumby, C. M., Rosen, J. L., & Roth, D. (1999). *Snow user guide* (Tech. Rep.). Champaign, IL, USA.

Chang, C.-C., & Lin, C.-J. (2001). *LIBSVM: a library for support vector machines.* (Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`)

Collins, M., & Duffy, N. (2001). Convolution kernels for natural language. In *Advances in neural information processing systems 14* (pp. 625–632). MIT Press.

Collins, M. J. (1999). *Head-driven statistical models for natural language parsing*. Unpublished doctoral dissertation, Philadelphia, PA, USA. (Supervisor-Marcus, Mitchell P.)

Cortes, C., & Vapnik, V. (1995). Support-vector networks. *Machine Learning*, *20*(3), 273–297.

Dang, H. T., Kelly, D., & Lin, J. (2008). Overview of the TREC 2007 question answering track. In *Proceedings trec 2007.*

Fellbaum, C. (Ed.). (1998). *WordNet: An electronic lexical database*. MIT Press.

Finkel, J. R., Grenager, T., & Manning, C. (2005). Incorporating non-local information into information extraction systems by gibbs sampling. In *Acl '05: Proceedings of the 43rd annual meeting on association for computational linguistics* (pp. 363–370). Morristown, NJ, USA: Association for Computational Linguistics.

Forner, P., Forascu, C., Moreau, N., Osenova, P., Prokopidis, P., & Rocha, P. (2008). Overview of the clef 2008 multilingual question answering track. In C. P. et al. (Ed.), *Working notes for the clef 2008 workshop.* Berlin, Heidelberg: Springer-Verlag.

Google. (2008, July). *We knew the web was big ...* `http://googleblog.blogspot.com/2008/07/we-knew-web-was-big.html`; Last accessed: 06/10/2009.

Gusfield, D. (1997). *Algorithms on strings, trees, and sequences - computer science and computational biology*. Cambridge University Press.

Hermjakob, U., Hovy, E., & Lin, C.-Y. (2002). Automated question answering in webclopedia: a demonstration. In *Proceedings of the second international conference on human language technology research* (pp. 370–371). San Francisco, CA, USA: Morgan Kaufmann Publishers Inc.

Hirschman, L., & Gaizauskas, R. (2001). Natural language question answering: the view from here. *Nat. Lang. Eng.*, *7*(4), 275–300.

Hsu, C., & Lin, C. (2001). *A comparison on methods for multi-class support vector machines* (Tech. Rep.). Department of Computer Science and Information Engineering, National Taiwan University, Taipei, Taiwan.

Huang, Z., Thint, M., & Qin, Z. (2008). Question classification using head words and their hypernyms. In *Emnlp* (p. 927-936).

Joachims, T. (1997). *Text categorization with support vector machines: Learning with many relevant features* (Tech. Rep. No. 23). Universität Dortmund, LS VIII-Report.

Judge, J., Cahill, A., & Genabith, J. van. (2006). Questionbank: creating a corpus of parse-annotated questions. In *Acl-44: Proceedings of the 21st international conference on computational linguistics and the 44th annual meeting of the association for computational linguistics* (pp. 497–504). Morristown, NJ, USA: Association for Computational Linguistics.

Jurafsky, D., & Martin, J. H. (2008). *Speech and language processing (2nd edition) (prentice hall series in artificial intelligence)* (2 ed.). Prentice Hall.

Katz, B. (1988). *Using english for indexing and retrieving* (Tech. Rep.). Cambridge, MA, USA.

Katz, B. (1997). Annotating the world wide web using natural language. In *Proceedings of the 5th riao conference on computer assisted information searching on the internet (riao '97).*

Kohavi, R. (1995). A study of cross-validation and bootstrap for accuracy estimation and model selection. In (pp. 1137–1143). Morgan Kaufmann.

Krishnan, V., Das, S., & Chakrabarti, S. (2005). Enhanced answer type inference from questions using sequential models. In *Hlt '05: Proceedings of the conference on human language technology and empirical methods in natural language processing* (pp. 315–322). Morristown, NJ, USA: Association for Computational Linguistics.

Kwok, C. C. T., Etzioni, O., & Weld, D. S. (2001). Scaling question answering to the web. In *Www '01: Proceedings of the 10th international conference on world wide web* (pp. 150–161). New York, NY, USA: ACM.

Li, F., Zhang, X., Yuan, J., & Zhu, X. (2008, August). Classifying what-type questions by head noun tagging. In *Proceedings of the 22nd international conference on computational linguistics (coling 2008)* (pp. 481–488). Manchester, UK: Coling 2008 Organizing Committee.

Li, X., & Roth, D. (2002). Learning question classifiers. In *Proceedings of the 19th international conference on computational linguistics* (pp. 1–7). Morristown, NJ, USA: Association for Computational Linguistics.

Lita, L. V., Hunt, W. A., & Nyberg, E. (2004). Resource analysis for question answering. In *Proceedings of the acl 2004 on interactive poster and demonstration sessions* (p. 18). Morristown, NJ, USA: Association for Computational Linguistics.

Littlestone, N. (1987). Learning quickly when irrelevant attributes abound: A new linear-threshold algorithm. In *Sfcs '87: Proceedings of the 28th annual symposium on foundations of computer science (sfcs 1987)* (pp. 68–77). Washington, DC, USA: IEEE Computer Society.

Mendes, A., Coheur, L., Mamede, N. J., Ribeiro, R. D., Matos, D. M. de, & Batista, F. (2008, September). Qa@l2f, first steps at qa@clef. *5152*.

Metsis, V., Androutsopoulos, I., & Paliouras, G. (2006). *Spam filtering with naive bayes – which naive bayes?*

Mitchell, T. (1997). *Machine learning*. McGraw-Hill Education (ISE Editions).

Moldovan, D., Paşca, M., Harabagiu, S., & Surdeanu, M. (2003). Performance issues and error analysis in an open-domain question answering system. *ACM Trans. Inf. Syst.*, *21*(2), 133–154.

Moschitti, A., & Basili, R. (2006). A tree kernel approach to question and answer classification in question answering systems. In *Lrec* (p. 22-28).

Pan, Y., Tang, Y., Lin, L., & Luo, Y. (2008). Question classification with semantic tree kernel. In *Sigir '08: Proceedings of the 31st annual international acm sigir conference on research and development in information retrieval* (pp. 837–838). New York, NY, USA: ACM.

Petrov, S., & Klein, D. (2007, April). Improved inference for unlexicalized parsing. In *Human language technologies 2007: The conference of the north american chapter of the association for computational linguistics; proceedings of the main conference* (pp. 404–411). Rochester, New York: Association for Computational Linguistics.

Porter, M. F. (1980). An algorithm for suffix stripping. *Program*, *14*(3), 130–137.

Quarteroni, S., & Manandhar, S. (2009). Designing an interactive open-domain question answering system. *forthcoming,Journal of Natural Language Engineering,Volume 15 Issue 1*.

Ravichandran, D., & Hovy, E. (2001). Learning surface text patterns for a question answering system. In *Acl '02: Proceedings of the 40th annual meeting on association for computational linguistics* (pp. 41–47). Morristown, NJ, USA: Association for Computational Linguistics.

Rifkin, R., & Klautau, A. (2004). In defense of one-vs-all classification. *J. Mach. Learn. Res.*, *5*, 101–141.

Samuel, A. L. (1959). Some studies in machine learning using the game of checkers. *IBM Journal of Research and Development*, *3*(3), 210–229.

Sharada, B. A., & Girish, P. M. (2004). *Wordnet has no 'recycle bin'*.

Simmons, R., Klein, S., & McConlogue, K. (1964). Indexing and dependency logic for answering english questions. 196–204.

Simmons, R. F. (1965). Answering english questions by computer: a survey. *Commun. ACM*, *8*(1), 53–70.

Solorio, T., Manuel Pérez-Couti n., Gémez, M. M. y, Luis Villase n.-P., & López-López, A. (2004). A language independent method for question classification. In *Coling '04: Proceedings of the 20th international conference on computational linguistics* (p. 1374). Morristown, NJ, USA: Association for Computational Linguistics.

Solorio, T., Pérez-Coutiño, M. A., Gómez, M. M. y, Pineda, L. V., & López-López, A. (2005). Question classification in spanish and portuguese. In *Cicling* (p. 612-619).

Soubbotin, M. M. (2001). Patterns of potential answer expressions as clues to the right answers. In *In proceedings of the tenth text retrieval conference (trec* (pp. 293–302).

Trotta, J. (2000). *Wh-clauses in english: Aspects of theory and description*. Kenilworth: Rodopi.

Turmo, J., Ageno, A., & Català, N. (2006). Adaptive information extraction. *ACM Comput. Surv.*, *38*(2), 4.

Vallin, A., Magnini, B., Giampiccolo, D., Aunimo, L., & Ayache, C. (2006). Overview of the clef 2005 multilingual question answering track. In C. Peters (Ed.), *Accessing multilingual information repositories.* Berlin, Heidelberg: Springer-Verlag.

Voorhees, E. M. (1999). The trec-8 question answering track report. In *In proceedings of trec-8* (pp. 77–82).

Voorhees, E. M. (2001). Question answering in trec. In *Cikm '01: Proceedings of the tenth international conference on information and knowledge management* (pp. 535–537). New York, NY, USA: ACM.

Voorhees, E. M. (2004). Overview of the trec 2004 question answering track. In E. M. Voorhees & L. P. Buckland (Eds.), *Trec* (Vol. Special Publication 500-261). National Institute of Standards and Technology (NIST).

Wang, Y.-C., Wu, J.-C., Liang, T., & Chang, J. S. (2005). Web-based unsupervised learning for query formulation in question answering. In *Ijcnlp* (p. 519-529).

W.A.Woods, Kaplan, R., & Webber, B. (1972). The lunar sciences natural language information system: Final report. In (Vol. BBN Report 2378). Cambridge, Massachussets: Bolt Beranek and Newman Inc.

Wikipedia. (2009, October). *Wikipedia.* `http://en.wikipedia.org/wiki/Wikipedia`; Last accessed: 06/10/2009.

Yarowsky, D. (1995). Unsupervised word sense disambiguation rivaling supervised methods. In *In proceedings of the 33rd annual meeting of the association for computational linguistics* (pp. 189–196).

Zhang, D., & Lee, W. S. (2003). Question classification using support vector machines. In *Sigir '03: Proceedings of the 26th annual international acm sigir conference on research and development in informaion retrieval* (pp. 26–32). New York, NY, USA: ACM.