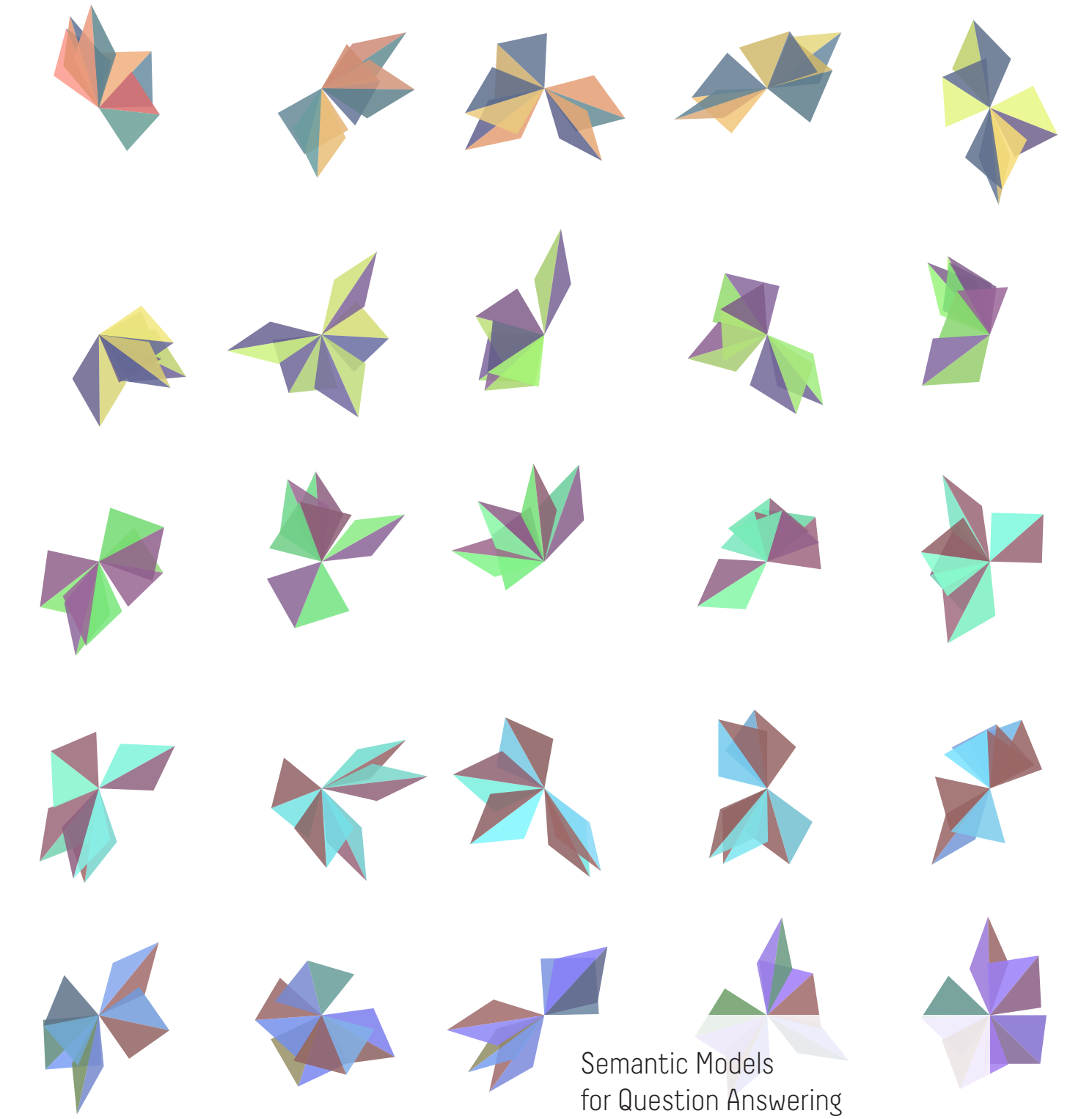


Semantic Models for Question Answering

Piero Molino

Semantic Models for Question Answering

Piero Molino
Università degli Studi di Bari Aldo Moro
Dipartimento di Informatica
Dottorato in Informatica XXVII ciclo

S.S.D.: INF/01

Supervisor: Dott. Pasquale Lops
Coordinator: Prof. Donato Malerba

A dissertation submitted in partial fulfillment
of the requirements for the degree of
Doctor of Philosophy in Computer Science

Final Exam 2015

Submitted *February 2015*

Cover design by Nicolò Loprieno (FF3300)
Graphics generated with Rosemantic by Piero Molino
https://github.com/w4nderlust/Rosemantic

This thesis is dedicated to my dad,
who always supported me during my university years.
This closes the circle.

Contents

List of Figures

List of Tables

List of Algorithms

List of Abbreviations

ABAN	Asker Best-Answerer Network
AI	Artificial Intelligence
ARN	Asker Replier Network
AS	Answer Scoring
CBEN	Competition-Based Expertise Network
CQA	Community Question Answering
DSM	Distributional Semantic Model
ES	Exact Substring
IR	Information Retrieval
LCS	Longest Common Subsequence
LR	Logistic Regression
LSA	Latent Semantic Analysis
ML	Machine Learning
MRR	Mean Reciprocal Rank
NDCG	Normalized Discounted Cumulative Gain
NLP	Natural Language Processing
QA	Question Answering
RF	Random Forests
RI	Random Indexing
SVD	Singular Value Decomposition
TL	Title Levenshtein
TTM	Term-Term co-occurrence Matrix
WWBM	Who Wants to Be a Millionaire?

Acknowledgments

"Mostly it is loss which teaches us about the worth of things."
– Arthur Schopenhauer, *Parerga and Paralipomena*

This thesis ratifies the end of a big chunk of my life.

As one could expect, I have mixed feelings about it, combining melancholy for the time spent in university with excitement for the new experiences to come. Maybe this is what it means to grow up, to realize that there's never black or white, but everything is made up of different aspects and there are just several shades of grey.

In the last years I had a lot trouble and many times I felt discouraged and powerless. But in the end my willpower prevailed and I overcome most of the problems I encountered. Only now, at the end of this path, I realize that all what I went through left a lot inside me and let me became a richer person and more conscious of my limits and my strengths.

In this process I encountered many people that helped me out with precious help during bad times. Maria, Paola, Antonio and Gabriella have been invaluable, but there are several others. They are too many to mention, but I want to thank them all even if they will never read these words.

Some other people actually helped me out during my PhD and in my research and in writing my thesis, and those are the one I would like to thank.

I won't thank Pasquale Lops, my supervisor, because Umberto Eco says it's tacky, but I would like to thank all the members of the SWAP research group for their support, in particular Pierpaolo and Annalina, that helped me become a better researcher and shared most of my research interests.

I would also like to thank all the people in the LACAM lab, always available for a chat and a smile.

The most formative experience I went through these years was the internship at Yahoo! Labs Barcelona, and it would not have been possible without the kindness and support of Luca Aiello. I also would like to thank all the people in the lab that made that experience unforgettable and in particular Alejandro Jaimes and Ricardo Baeza-Yates who gave me the opportunity.

Some insights in this thesis stemmed from discussion I had with a lot of great

researchers from all around the world, who kindly shared their opinion on my work during conferences and summer schools. The ones I would like to thank the most are W. Bruce Croft, Giuseppe Attardi, Hugo Zaragoza, Massimiliano Ciaramita, Julio Gonzalo, Hui Fang, Bernardo Magnini, John Tait, Liadh Kelly and Roberto Navigli.

Finally, one of the things I will keep with me forever of this PhD period are the fantastic friends I made from all around the world, meeting them was probably the most valuable thing pursuing a PhD enabled me to do and their enthusiasm gave me the strengh to carry on, so thank you all.

Abstract

In this thesis we investigate the possibility to adopt distributional semantic representations for Question Answering. We propose a way to represent questions and answers and calculate their semantic relatedness with different Distributional Semantic Models. We furthermore combine the semantic relatedness with several other criteria using a Learning to Rank approach in order to exploit the additional information they bring and assess their usefulness. This method has been tested on large-scale real-world settings of European Union legislation and social Q&A communities (Yahoo! Answers) on the task of passage retrieval and best answer prediction with substantial improvement over state-of-the-art baselines. We tested our approach in a language game application, building an artificial player for "Who Wants to Be a Millionaire?" tv quiz show capable of outperforming human players both in the answering accuracy and in the ability to play the game.

1

Introduction

> *"The Answer to the Great Question... Of Life, the Universe and Everything... Is... Forty-two," said Deep Thought, with infinite majesty and calm.*
>
> – Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

In a world where everyone is always connected and information is always available, people do not need to keep everything in memory. This enables the possibility to learn new things and suggest a shift in the attention from search technologies to technologies that actually help us find something new, have new insights and connect pieces of knowledge. The most natural and straightforward way for a human being to do so is asking questions to other people, in natural language. Language is the main way human have to share their knowledge and is probably the most important technology humans have been able to build. It is a unique characteristic of human beings and is able to shape our mind and influence our perception and interpretation of reality. Studying and understanding language not only lets us dive deeper in our own learning process, but makes us learn about ourself.

Using language to ask and answer questions is a "too human" activity and this is why trying to create a computer program capable of doing it is so challenging and fascinating, it is the same fascination at the very bottom of every Artificial Intelligence (AI) study. The implications of an AI fully capable of understanding the meaning of questions and give convincing answers would be groundbreaking and will force us to revise our definitions of intelligence and of human race.

This kind of AI will also be an incredible "tool for thought" and a fantastic learning companion used by everyone for pursuing their knowledge and make new discoveries. It would probably redefine the idea of intelligence as we know it and be the best tool humans have for knowledge enrichment. This is also the direction most commercial search engines are heading to: creating an intelligent

assistant to help you search for things. They are trying to build true Question Answering (QA) systems, capable of answering natural language questions. In February 2011, IBM's Watson supercomputer powered by DeepQA technology was able to beat the two highest ranked players of the quiz show "Jeopardy!" [Ferrucci et al., 2013, 2010], and it was a strong display of what this kind of technology is actually capable of.

1.1 Research Outline

This thesis is a contribution towards building such AI. It focuses mainly on one aspect: the representation of meaning. Being able to represent the meaning of both questions and answers makes it possible to match them more precisely and in the end makes it possible to find more accurate answers.

The way to represent meaning we decided to adopt is mapping language to geometrical spaces built observing how words occur with each other inside texts. This approach is called Distributional Semantics and arises from the study of context, in particular from Wittgenstein's argument that "meaning is use" and words are not defined by reference to the objects they designate, nor by the mental representations one might associate with them, but by how they are used [Wittgenstein, 1953].

The most important task for a QA system is the ranking of the answers to a specific question and we will propose several criteria based on Distributional Semantics for ranking candidate answers.

Doing so, the research questions we want to answer are:

- *RQ1. Are the distributional semantic representations good representations for the meaning of questions and answers?*
  We address this question building a QA system that exploits this representation and testing it on two big scale problems.

- *RQ2. Can distributional semantic representations be combined with other criteria in order to obtain a better ranking of the answers?*
  In order to answer this question we propose to combine Distributional Semantics with several other criteria for answer ranking in a Machine Learning setting. We experimented on huge scale dataset the combination of criteria and the contribution of each criterion family to the quality of the ranking.

We also wanted to investigate if the QA technology we built is capable of being employed in real world scenarios, leveraging common sense knowledge and competing with humans, in a way that does not depend on a specific language.

This arises two other research questions:

- *RQ3. To what extent can a QA system be designed in a language-independent way, by preserving its effectiveness?*
  We cope with this question by assessing the effectiveness of a QA and answer scoring framework for English and Italian. The QA framework leverages Wikipedia and DBpedia open knowledge sources, while the answer scoring module supplies several criteria to score candidate answers and to effectively combine scores through machine learning techniques.

- *RQ4. Is it possible to develop an artificial player for the WWBM game able to outperform human players?*
  We address this question by comparing the accuracy of the human players against that of an artificial player built using the QA framework in playing "Who Wants to Be a Millionaire?" (WWBM). We evaluate the ability of the artificial player to play the WWBM game with all its rules, i.e. usage of "lifelines", answering in a condition of uncertainty, retiring from the game by taking the earned money.

1.2 Main Contributions

The contribution of this thesis is three-fold: theoretical, empirical and algorithmic.

Theoretical contribution

- The first Distributional-Semantics-based approach for representing questions and answers
- A model for ranking answers based on Distributional Semantics
- A general language-independent QA architecture
- A new answer scoring approach that exploits an high number of criteria
- The architecture of an AI capable of beating humans in playing WWBM

Empirical contribution

- An empirical evaluation of the proposed Distributional-Semantics-based approach
- The most large scale evaluation of different answer ranking criteria yet
- The evaluation of answer ranking criteria on different question types
- A comparison between a QA system and humans on answering WWBM questions

- A comparison between a QA system and humans on playing WWBM games

Algorithmic contribution

- A decision making algorithm for playing WWBM

A further contribution of this thesis is the creation of a WWBM dataset made freely available (see Chapter 7).

1.3 Thesis Overview

The thesis is organized as follows.

In the next chapter (Chapter 2) an overview of the QA field is given, with description of Open Domain and Closed Domain systems and the distinction of Factoid and non-Factoid QA. The chapter contains also the description of our proposed QA architecture.

In Chapter 3, we give an overview of Distributional Semantics and we describe several models we adopt. The chapter contains also the description of our proposed Distributional-Semantics-based representations and the ranking model.

In Chapter 4, we describe the Learning to Rank setting we adopt for combining different ranking criteria, giving also an overview of the main approaches and describing the algorithm we adopt.

In Chapter 5, we show the results of a passage retrieval experiment carried out on the *ResPubliQA 2010 Dataset*. The aim is to test the effectiveness of our proposed method of integration of Distributional Semantic Models in a QA system.

In Chapter 6, we describe an extended experiment carried out on two massive real-world datasets extracted from Yahoo! Answers. We give an overview of the community / social QA and expert finding research and we describe a huge variety of features to combine our proposed ones to.

In Chapter 7, we propose our artificial player for WWBM. We describe the history of AIs playing language games, we propose our architecture and scoring criteria, a strategy for decision making, and we compare against humans both in terms of accuracy in answering questions and ability to play the game.

Finally we draw conclusions in Chapter 8 answering the research questions.

1.4 Origins

The following publications form the basis of chapters in this thesis.

- Chapter 2 is based on [Molino and Basile, 2012].

- Chapter 3 is based on [Molino et al., 2012] and [Molino and Aiello, 2014].
- Chapter 4 is based on [Molino and Aiello, 2014].
- Chapter 5 is based on [Molino et al., 2012].
- Chapter 6 is based on [Molino and Aiello, 2014] and on the journal paper to appear named "Expert Finding meets Distributional Semantics for Social Question Answering".
- Chapter 7 is based on [Molino et al., 2013a] and on the journal paper to appear named "Playing with Knowledge: A Virtual Player for "Who Wants to Be a Millionaire?" that Leverages Question Answering Techniques".

Finally, this thesis draws from insights and experiences gained in [Molino et al., 2013b, Molino, 2013].

Part I

Semantics in Question Answering

2

Question Answering

"Maybe the only significant difference between a really smart simulation and a human being was the noise they made when you punched them."
– Terry Pratchett, *The Long Earth*

In this chapter we give an overview of the Question Answering field, with description of Open Domain and Closed Domain systems and the distinction of Factoid and non-Factoid QA. The chapter contains also the description of our proposed QA architecture.

2.1 Intelligent Systems to Answer Questions

Question Answering (QA) emerged in the last decade as one of the most promising fields in Artificial Intelligence, as highlighted by the organization of several competitions in international conferences [Voorhees and Tice, 1999, Peñas et al., 2010], but the first studies can be dated back to 1960s [Green et al., 1961, Simmons, 1965].

The task of QA is to find correct answers to users' questions expressed in natural language. This is carried out exploiting techniques borrowed from Information Retrieval (IR) and Natural Language Processing (NLP) and Machine Learning (ML). Differently from search engines, which output a lists of full-text documents that users have to check in order to find the needed information, QA systems are able to answer users' questions with answers that could range from exact facts, dates, names, places, to passages of text like descriptions, summaries or explanations.

In recent years some enterprise applications have shown the potential of the state-of-the-art QA technology, such as the IBM's Watson/DeepQA [Ferrucci et al., 2010, Ferrucci, 2011]. This system was able to outperform the human champions of the popular American TV quiz "Jeopardy!", a game that requires

vast knowledge, common sense and language understanding. This kind of goal would be considered extremely difficult to achieve just ten years ago.

2.2 Background

QA systems are usually classified depending on the kind of data they are able to access and the kind of questions and answers they are able to return to users.

2.2.1 Closed vs. Open domain

Closed-domain QA refers to systems working in specific and limited domains (such as medicine or finance). Dealing with questions in a Closed-domain QA is generally an easier task since some kind of domain-specific knowledge can be exploited, and only a limited type of questions are accepted, such as those asking for descriptive rather than procedural information.

On the other hand, *Open-domain QA* does not refer to a specific domain and deals with more general questions. Dealing with questions in an Open-domain QA generally requires the use of world knowledge to search for an answer.

The Web is generally used as a source of knowledge and the redundancy of information is exploited as a signal for the quality of the answer, for example selecting the answers according to their frequency among the search results [Dumais et al., 2002, Lin, 2007]. This technique is often complemented with textual pattern extraction and matching to find the exact answers and rank them by confidence [Harabagiu et al., 2000a, Paşca and Ribarov, 2004].

2.2.2 Factoid vs. non-Factoid

Another classification of QA systems distinguishes between *factoid QA* and *non-factoid QA*. The former deals with questions whose answers are usually named entities (names of persons, organizations, locations) or facts (time expressions, quantities, monetary values, etc), while the latter focuses on causation and reason questions, and the expected answers have the form of passages of text (sentences, groups of sentences, paragraphs or short texts).

The passage retrieval step is, anyway, fundamental in both factoid and non-factoid QA as in the former the answers are extracted from the obtained passages, while in the latter the passage corresponds to the candidate answer itself, even if the passage for non-factoid QA is much longer, as shown in [Verberne et al., 2008].

Factoid QA received wide attention and a variety of different approaches and systems are described in literature. For the sake of brevity we describe only the most prominent approaches. Factoid QA systems heavily rely on information extraction techniques, including the adoption of linguistic patterns to identify

the specific answer. For Closed-domain QA, QA systems working in specific and delimited domains with a relatively small set of documents, NLP methods are always used for a deeper understanding of users' questions and for the matching of passages extracted from documents [Harabagiu et al., 2000b, Hovy et al., 2000]. The most commonly adopted linguistic analysis steps include: stemming, lemmatization with dictionaries, part-of-speech tagging, parsing, named entity recognition, lexical semantics (Word Sense Disambiguation) and semantic role labeling. Their adoption plays a key role, as stressed in [Chen et al., 2001, Moldovan et al., 2003], since there is likely to be really few answers to users' questions and the way in which they are expressed may be significantly different from the question. Furthermore, they are helpful also for question classification [Li and Roth, 2006]. Thus NLP is essential for uncovering complex lexical, syntactic, or semantic relationships between questions and candidate answers.

Open-domain QA systems for factoid QA adopt the Web as source of knowledge, so they can exploit the redundancy of this data source, selecting the answers according to their frequency among the web search results [Dumais et al., 2002, Lin, 2007]. Alongside with this technique, textual pattern extraction and matching is used to find the exact answers and to rank them by confidence [Harabagiu et al., 2000a, Paşca and Ribarov, 2004]. We don't use redundancy based techniques as we will be dealing with selected and highly accurate source of information that do not usually contain more than few entries of the correct answer.

In the last few years non-factoid QA received more attention. It focuses on causation, manner and reason questions, where the expected answer has the form of a passage of text. Depending on the structure of the corpus, the passages can be single sentences, groups of sentences, paragraphs or short texts.

The presence of annotated corpora made available for the competitions inside the Text REtrieval Conference (TREC) [Voorhees and Tice, 1999] and Cross Language Evaluation Forum (CLEF) [Peñas et al., 2010], alongside with different sources, such as hand-annotated answers from Wikipedia [Verberne et al., 2010], hand-built corpora [Higashinaka and Isozaki, 2008], Frequently Asked Questions lists [Soricut and Brill, 2006, Agichtein et al., 2008] and Yahoo! Answers Extracted corpus [Surdeanu et al., 2011], allows to use ML techniques to tackle the problem of ranking the passages for further extraction in factoid QA [Agarwal et al., 2012].

In particular, Learning to Rank (MLR) [Liu, 2009] algorithms are used in order to output a sensible ranking of the candidate answers. MLR algorithms apply ML techniques to the problem of ordering a set of items depending on the queries. In the QA case the items are answers and the queries are the questions. Usually, the features for the learning task are different similarity measures between the query and the item. In the IR tasks TF-IDF, BM25 and

Language Modeling based features are often used. In [Verberne et al., 2008] the adoption of linguistically motivated features is shown to be effective for the QA task, while in [Verberne et al., 2011] different MLR algorithms were compared over the same set of features.

In [Punyakanok et al., 2004] the authors adopted a distance function for calculating the similarity of question and answers that calculates an approximate tree matching of their parse trees. The idea is expanded in [Shen and Joshi, 2005], where the authors trained dependency tree kernels to compute similarity in a supervised fashion. In [Sun et al., 2005] both syntactic and semantic parsing are adopted in order to improve matching. The Ephyra framework [Schlaefer et al., 2007] leverages semantic role labeling to identify semantic structures in documents that match those in the question. Predicate-argument structures taken from semantic role labeling built from questions and expected answers have been also shown to be useful for the task, as reported in [Bilotti, 2010]. However, semantic approaches can also be exploited to provide an enriched visual representation of the answer in the form of a semantic graph, as described by [Dali et al., 2009].

The exploitation of structural and semantic information has been shown to help improve answer ranking. In [Severyn and Moschitti, 2012] the adoption of tree kernels over shallow parse structures helps in obtaining a significant improvement in answer ranking. WordNet synsets are used for expansion and comparison in [Verberne et al., 2011], while in [Higashinaka and Isozaki, 2008] a wide range of semantic features is considered: by exploiting WordNet and gazetteers, semantic role labeling and extracted causal relations, the authors obtained accurate results in answering Why-questions. A comprehensive large scale evaluation, alongside with the introduction of new features based on translation models and web correlation, was carried out in [Surdeanu et al., 2011], where the authors also adopt super-senses (coarse grained WordNet concepts) as lexicalization level.

We adopt similar linguistic techniques as the cited ones, but with several important differences. In particular, for the syntax analysis we rely on dependency parsing in order to allow real-time question analysis and answer re-ranking, while for semantics, we adopt Distributional Semantic Models (DSMs) rather than knowledge based approaches. This is novel in the QA field, especially for answer re-ranking. DSMs allow us to calculate semantic similarity independently of lexicons and independently from the language of the question and the documents.

Machine translation has been applied to factoid QA in order to automatically learn question transformations. In [Echihabi and Marcu, 2003] the authors adopted IBM model 4 [Brown et al., 1993] for the task, including lexical, parse-tree and named entities features. An alternative approach is presented in [Cui

et al., 2005], where only the most significant words are aligned in the translation model and the similarity of the dependency paths of questions and answers are computed as mutual information of correlations.

Statistical models have been applied to the task of non-factoid QA: in [Soricut and Brill, 2006] the authors extracted QA pairs from FAQs obtained from the Web in an unsupervised manner. They show how different statistical models may be used for the problems of answer ranking, selection and extraction.

The importance of understanding how questions transform into answers has been investigated deeply. In [Agichtein et al., 2001] lexical transformations learned from questions and web queries pairs help in retrieving good candidate answers for both factoid and non-factoid QA. Other evidence of the usefulness of translation models in improving factoid QA is given in [Murdock and Croft, 2005] and [Xue et al., 2008]. Machine translation has been used also as a query expansion model successfully applied to FAQ retrieval in [Riezler et al., 2007].

In our work we adopt IBM Model 2 [Brown et al., 1993] for single word translations from the "question language" to the "answer language" and exploit the probability of the translation as a MLR feature.

2.3 Architecture

In order to answer our research questions, we built a QA framework distinctively focused on non-factoid QA that implements most of the characteristics that were proposed in literature. This enables us to add DSMs to a well established set of features so that the performance gain obtainable with those models can be directly put in comparison with strong baselines.

The built framework is called QuestionCube and it is a multilingual QA framework built using NLP and IR and MLR techniques [Molino and Basile, 2012, Molino et al., 2012].

The main aim of the architecture, shown in Figure 2.1, is to enable the creation of a QA system simply by dynamical composition of components, as suggested in [Schlaefer et al., 2006].

The high level of abstraction of the components allows to add support for a new language just by creating new interchangeable analyzers which implement the algorithms for that specific language.

The system works in two separate steps. At indexing time, the system builds two different indices, one for documents and one for passages belonging to each document. They contain the original text alongside with the linguistic annotations coming from the NLP pipeline.

At query time, the user's question q is analyzed by a NLP pipeline. The result of this pipeline is a text tagged with linguistic annotations useful for a deeper representation and deeper question-answer matching. The question q

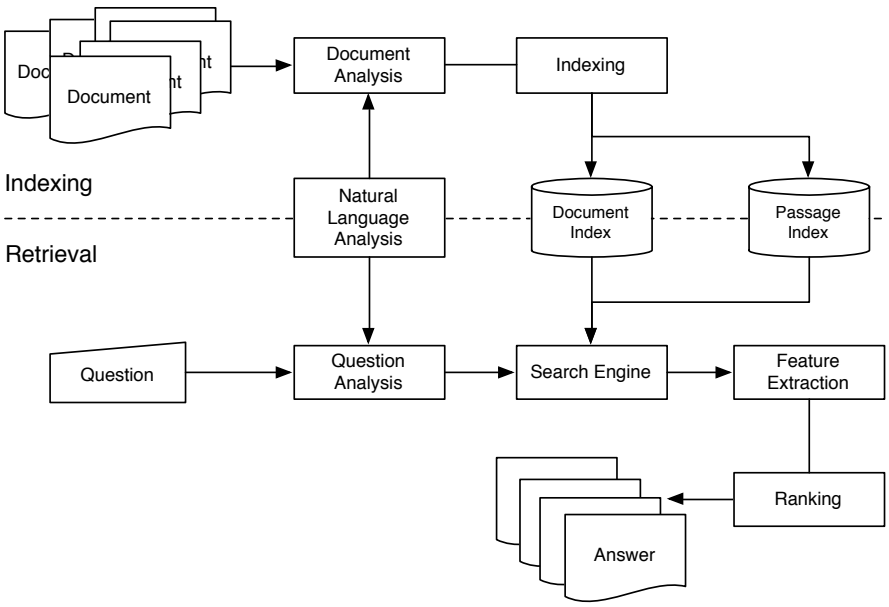


Figure 2.1: QuestionCube architecture

is then passed to the search engines, from which a list of candidate answers containing passages p is obtained.

Then, the feature extraction pipeline is responsible for extracting features f_p from the candidate answers passages p retrieved by search engines. Those features f_p are adopted by the ranking component to output the ranked list of candidate answer passages p_{ranked} that is presented to the user.

2.3.1 Natural Language Analysis

The natural language analysis module consists of a pipeline of NLP analyzers and a data-structure to contain linguistic annotated text as shown in Figure 2.2. The NLP pipeline is easily configurable depending on the application domain of the QA system.

NLP analyzers are provided for both English and Italian. The stemmer is implemented with Snowball[1] both for English and Italian. The lemmatizer is realized by exploiting the morpho-syntactic analyzer of WordNet API [Fellbaum, 1998] for English, while Morph-it [Zanchetta and Baroni, 2005] is exploited for Italian. Named Entity Recognition (NER) is performed by a machine learning classifier based on Support Vector Machines [Cortes and Vapnik, 1995] using an open-source tool called YAMCHA [Kudo and Matsumoto, 2003]. The same tool is used for the chunker component. PoS tags and lemmas are adopted as features

[1] Available on-line: http://snowball.tartarus.org/

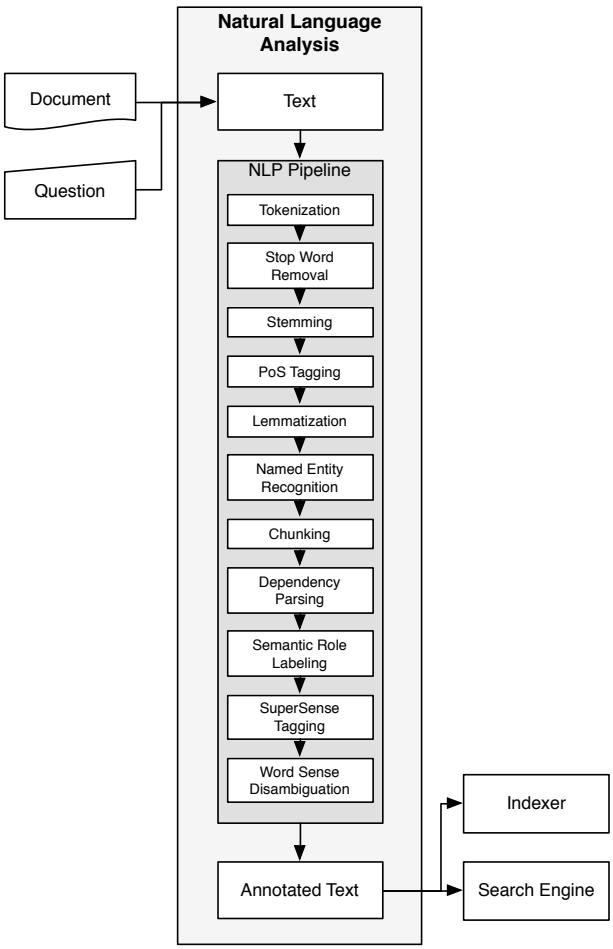


Figure 2.2: Natural language analysis module

for chunking and NER. The Word Sense Disambiguation (WSD) uses the UKB algorithm [Agirre and Soroa, 2009], which is a graph-based technique based on a personalized version of PageRank [Brin and Page, 1998] over WordNet graph.

Alongside with this components, wrappers to popular NLP libraries such as Stanford CoreNLP [Manning et al., 2014][2] and ClearNLP suite [Choi, 2012][3] have been developed, enabling dependency parsing and semantic role labeling for English.

Super-sense tagging, the task that assigns a WordNet coarse grained synset to the verbs, nouns and adjectives in a sentence, is carried out implementing the algorithm in the SuperSense Tagger proposed by Ciaramita and Altun [2006][4].

[2] `http://nlp.stanford.edu/software/corenlp.shtml`
[3] `http://clearnlp.com`
[4] `http://sourceforge.net/projects/supersensetag`

The output of the NLP analyzers is a set of tags that are added to the text representation of both passages p (at indexing time) and questions q (at query time). Those are passed respectively to the Indexer and the Search Engine.

2.3.2 Search Engine

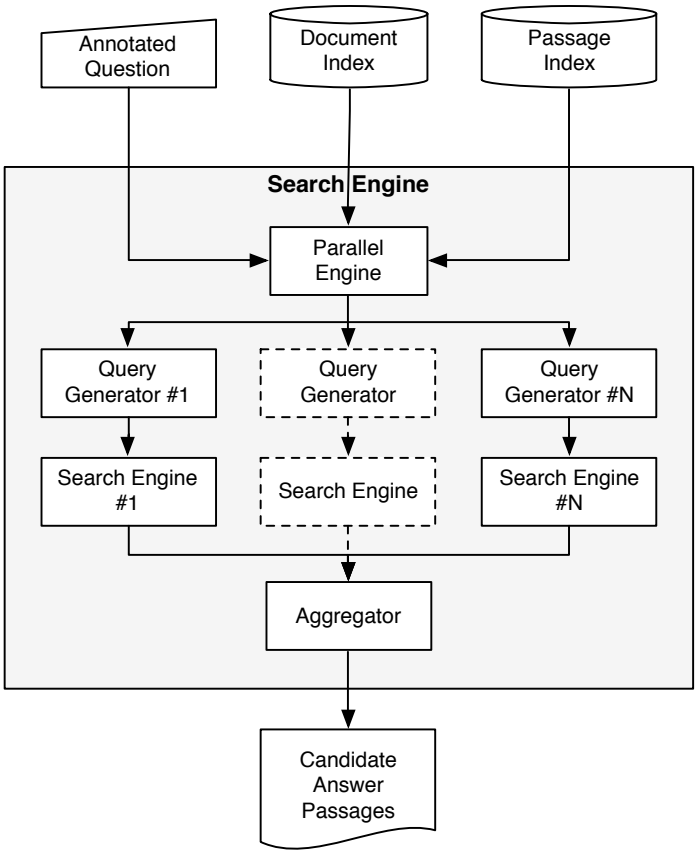


Figure 2.3: Search engine module

The search engine module is designed to allow the integration of several information retrieval strategies, and thus the aggregation of their results, as shown in Figure 2.3. Since the parallel searcher enables modularity, it is possible to add an arbitrary number of different search engines. When a new question q comes, the parallel searcher calls each engine and merges their outputs in a single list P.

The list P contains all the candidate answer passages p collected from all the engines, each one with a reference to the engines that retrieved it and the score assigned by each engine.

Each search engine has its own query generation component, as the query syntax may change among different engines. Moreover, each query generator can use different lexicalizations: terms, stems, lemmas, WordNet synsets and so on, to generate the query representation. This approach makes the framework highly modular and flexible and allows adding a new search engine inside the framework with minimal effort.

The main goal of using more than one search engine is to rely on several retrieval strategies in order to exploit different levels of lexicalization and different retrieval models.

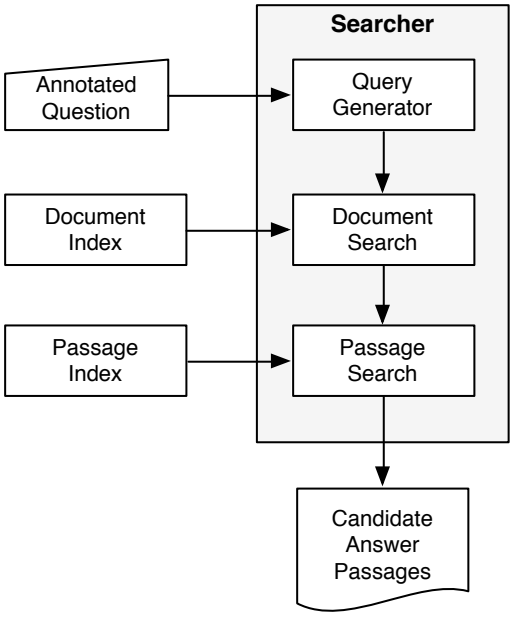


Figure 2.4: Single search engine

The process performed by each search engine se is described in Figure 2.4. Each query generator builds the query qe^{se} for its search engine from the linguistically annotated question q provided by the parallel engine. Moreover, the query generator may implement different query-improvement techniques (such as relevance feedback and query expansion). Hence, the search engine executes the query qe^{se} to return the list of best scored documents D^{se}. The passage index is used to obtain passages P^{se} that are part of the retrieved documents D^{se}. These passages P^{se} are merged into one single list P by an aggregation component and finally passed to the feature extraction pipeline.

The search engines implemented in the framework are based on Apache Lucene[5] and thus implement the *BM25* model [Robertson and Zaragoza, 2009],

[5]Available at http://lucene.apache.org/

TF-IDF [Salton et al., 1975]) based retrieval and *Language Modeling* (with Dirichlet priors [Zhai and Lafferty, 2001]). All three retrieval models can use different lexicalizations as tokens, depending on the linguistic annotations added to the question and the passages.

The query generation component for those searchers allows three different query-improvement techniques:

- *Query expansion* through WordNet synonyms of synsets found in the question;
- *Kullback-Leibler Divergence*, a statistical technique that exploits the distribution of terms in the top-ranked documents [Carpineto et al., 2001, Lv and Zhai, 2010];
- *Divergence From Randomness*, a statistical technique that weights the terms' distribution with the Bose-Einstein *Bo1* weighting scheme [Amati and van Rijsbergen, 2002].

The WordNet-based query expansion can be used only if the question has been disambiguated.

2.3.3 Feature Extraction

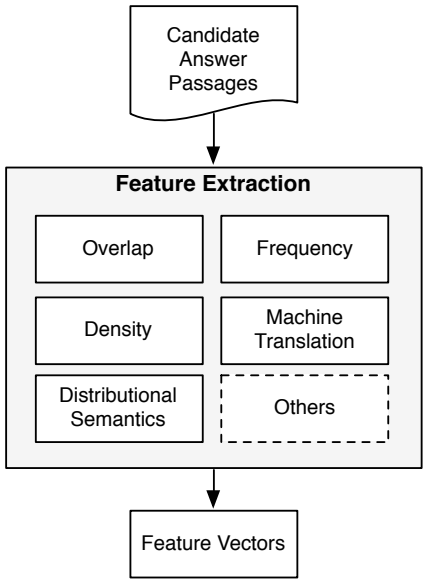


Figure 2.5: Candidate answers feature extraction module

The feature extraction module, sketched in Figure 2.5, contains a pipeline of feature extractor components that assigns a score to each candidate answer

leveraging its textual content and its metadata. At the end of the process, each candidate answer passage p_i carries a feature vector $\phi(p_i)$.

Here we present a small list of linguistic similarity features that are common in all the experiments that we will describe later. Each experiment can have its own additional feature set that will be motivated and described on a case-by-case basis.

- TERM OVERLAP: counts the ratio of terms in common between the question and the answer as $\frac{|t_q \cap t_a|}{|t_q|}$, where t_q is the set of terms belonging to the question and t_a the set of terms belonging to the answer.
- LEMMA+POS OVERLAP: counts the same formula of TERM OVERLAP but considering the concatenation of lemmas and PoS tags instead of simple terms.
- LEMMA+POS DENSITY: We adopted a slight modification of the Minimal Span Weighting proposed by Monz [2004] retaining only the local similarity part:

$$\left(\frac{\mid t_q \cap t_a \mid}{1 + \max(mms) - \min(mms)}\right)\left(\frac{\mid t_q \cap t_a \mid}{\mid q \mid}\right)$$

  where t_q and t_a are the sets of concatenations of lemmas and PoS tags respectively of the question and the answer; $\max(mms)$ and $\min(mms)$ are the initial and final location of the shortest sequence of answer concatenations containing all the question concatenations. More details in Section 6.3.1.
- EXACT SEQUENCE: calculates the exact overlap of the sequences of words in the question and the answer normalized by the length of the question.

For the full list of available features (225 different ones) and for a more detailed description, see Section 6.3.

2.3.4 Ranking

After the feature vector $\phi(p_i)$ of each candidate answer passage p_i has been extracted by the feature extraction module, the candidate answers passages P are passed to the ranking module that exploit those features to output a sensible ranking P_{ranked}. The process is sketched in Figure 2.6.

Depending on the availability of training data (questions with a list of correct answers) two scenarios should be considered: if there is such training data for the specified corpus or this training data can be gathered from the users, then a Learning to Rank model can be learned, otherwise an unsupervised strategy is available in order to provide a ranking.

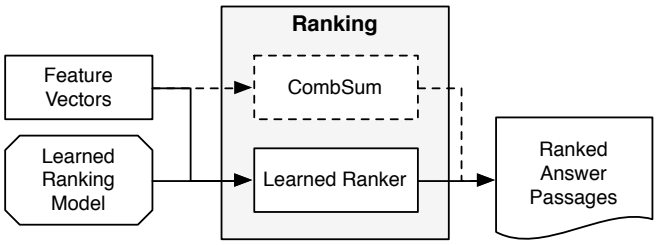


Figure 2.6: Ranking module

In case of availability of training data, the Learned Ranker component is adopted. Its training and its use at runtime are described in Chapter 4.

If there is no training data, the CombSum component is used. It implements the CombSum strategy [Fox and Shaw, 1993] that consists in first normalizing the values of each feature j in the feature vector $\phi(p_i)$ of each candidate answer passage p_i in P with the z-score normalization [Fox and Shaw, 1993]:

$$z_{i_j} = \frac{\phi(p_i)_j - \mu_j}{\sigma_j}$$

where μ_j in the average of $\phi(p)_j$, the values of the feature j for all candidate passages $p \in P$, and σ_j is the standard deviation of $\phi(p)_j$. Then all the normalized scored are combined by summing them to obtain an aggregated score s :

$$s_i = \sum_{j=1}^{|z|} z_{i_j}$$

The candidate answer passages p_i are finally ranked according to their summed score s_i that represents an overall indication of the quality of the answer according to all the considered features.

3

Semantics

"But if thought corrupts language, language can also corrupt thought."
– George Orwell, *1984*

In this chapter we give an overview of Distributional Semantics and we describe several models we adopt. The chapter contains also an overview on the adoption of semantic models in QA and the description of our proposed Distributional Semantics based representation and semantic similarity used as ranking model.

3.1 Distributional Semantic Models

Distributional Semantics Models (DSMs) have gained mass appeal in the joint area lying at the intersection between Computational Linguistics, Cognitive Science and Geometry. These models represent word meanings through contexts: different meanings of a word can be accounted for by looking at the different contexts in which the word occurs. Philosophical insight of distributional models can be ascribed to Wittgenstein's quote *"the meaning of a word is its use in the language"*[Wittgenstein, 1953]. The idea behind DSMs can be summarized as follows: if two words share the same linguistic contexts they are somehow similar in their meaning. For example, analyzing the sentences "drink a glass of wine" and "drink a glass of beer", we can assume that the words "wine" and "beer" have similar meaning because they appear in proximity of the same set of words (drink, a, glass, of).

This insight can be implemented with a geometrical representation of words as vectors in a *semantic space*. Each term is represented as a vector whose components are contexts surrounding it. In this way, the meaning of a term across a corpus is thoroughly conveyed by the contexts it occurs in, where some definitions of contexts may be the set of co-occurring words in a document,

in a sentence or in a window of surrounding terms. The word vector is built analyzing (e.g. counting) the contexts in which the term occurs inside a corpus. The resulting vector of each word will be closer to the vectors of words that have a similar meaning and this property can be exploited in order to deal with synonyms.

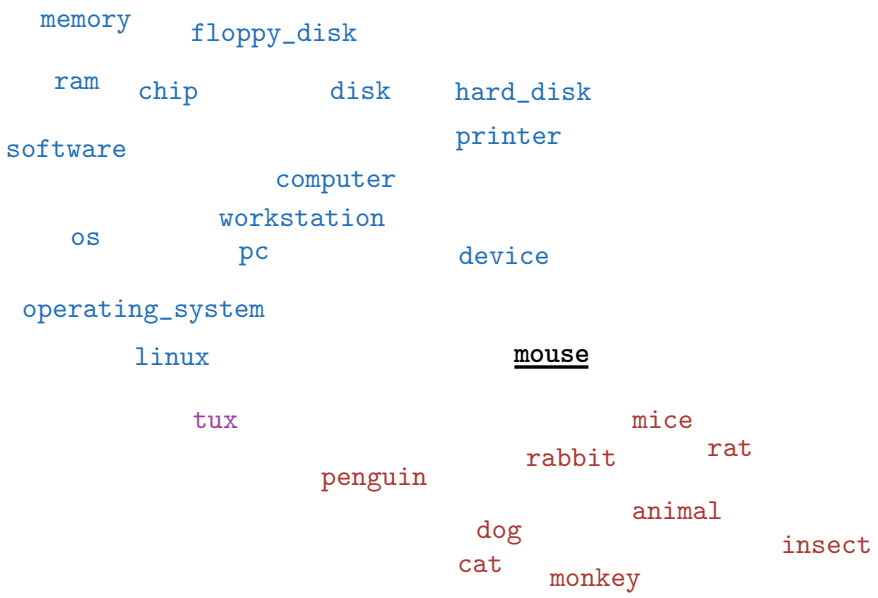


Figure 3.1: A (bidimensionally projected) depiction of the portion of semantic space around the word *"mouse"*

A (bidimensionally projected) depiction of the portion of semantic space around the word *"mouse"* is shown in Figure 3.1. Concepts that are semantically closer are also closer in the semantic space, like *"mice"* and *"rat"* or *"memory"* and *"ram"*, while distant concepts are far apart, like *"software"* and *"animal"* or *"memory"* and *"insect"*.

As for the vector representations, given the sentences "drink a glass of wine", "wine is made of grapes", "drink a glass of beer" and "beer is made of hops", and considering word occurring in the same sentence as context, we can represent the words "wine" and "beer" with the vectors shown in Figure 3.2, simply counting the number of occurrences.

Semantic spaces have important advantages over other textual features. They do not require specific text operations, only tokenization is always needed. They are also language-agnostic and independent of the specific corpus. This implies low computational cost and independence from any external source.

The earliest and simplest formulation of such a space has root in the Vector

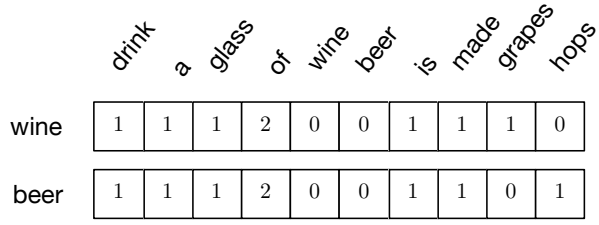


Figure 3.2: An example of vector representation of words in a DSM

Space Model [Salton et al., 1975], one early model in Information Retrieval. Since then, several linguistic and cognitive tasks have taken advantage from the use of semantic spaces, such as synonym choice in the Test of English as a Foreign Language (TOEFL) [Landauer and Dumais, 1997], semantic priming [Landauer and Dumais, 1997, Burgess et al., 1998, Jones and Mewhort, 2007], similarity of semantic relations [Turney, 2006, Turney and Littman, 2005], essay grading [Wolfe et al., 1998, Foltz et al., 1999], automatic construction of thesauri [Schütze and Pedersen, 1995] and word sense induction [Schütze, 1998]. DSMs have also been adopted to improve significantly many NLP applications [Basile, 2011, Collobert et al., 2011, Turian et al., 2010].

DSMs are also widely used to solve problems related to word similarity and semantic composition of meanings. A useful survey of the use of VSMs for semantic processing of text is reported in [Turney and Pantel, 2010], while an analysis of some compositional operators is described in [Mitchell and Lapata, 2010].

DSMs have not been used directly in QA, while some applications to IR exist. In [Widdows and Ferraro, 2008] the authors describe a tool which relies on RI to build an IR model. LSA is widely used in IR to perform term-doc matrix reduction obtaining good results and significant improvement with respect to the classical Vector Space Model [Deerwester et al., 1990]. In [Schütze and Pedersen, 1995] an approach to ambiguity resolution in IR is proposed. The authors describe a sense-based retrieval, a modification of the standard vector-space model, in which the meanings of a word are inferred by applying clustering technique to a word space. The similarity between the word vector and the closest cluster centroid will give the proper sense. The IR system proposed by the authors gives an improvement in precision with respect to the word-based retrieval. Moreover, in [Basile et al., 2011], an IR system able to combine word sense disambiguation and word sense discrimination based on RI is proposed. The combination of WSD and RI is performed by a semantic engine, SENSE, which is able to combine several document representations in a unique framework.

We aim at exploiting DSMs for performing Question Answering, a task to which they have never been applied before, exploring how to integrate them inside a preexistent QA system. Our insight is based on the observation of the capability of these spaces to capture paradigmatic relations between words. Exploiting those relations to calculate a measure of semantic similarity should result in a ranking of candidate answers based on their semantic relatedness with respect to the question asked. We think this kind of similarity will be a strong feature to adopt, as it will integrate a way of dealing with synonyms.

In the following of the chapter, we first describe how we build the semantics space, then we describe the adopted DSMs and finally we describe our strategy to adopt them inside our QA system.

3.1.1 Co-occurrence Matrix Construction

Our semantic spaces are modeled by a co-occurrence matrix. The linguistic context taken into account is a window w of co-occurring terms. In our experiments we adopt a window of size 5 centered on the current term. Given a reference corpus[1] and its vocabulary V, a $n \times n$ co-occurrence matrix is defined as the matrix $\mathbf{M} = (m_{ij})$ whose coefficients $m_{ij} \in \mathbb{R}$ are the number of co-occurrences of the words t_i and t_j within a predetermined distance w.

The $term \times term$ matrix $\mathbf{M}$, based on simple word co-occurrences, represents the simplest semantic space, called Term-Term co-occurrence Matrix (TTM).

An example $term \times term$ matrix $\mathbf{M}$ is shown in Figure 3.3. The corpus from which it is obtained are the same four sentences of Figure 3.2: "drink a glass of wine", "wine is made of grapes", "drink a glass of beer" and "beer is made of hops".

In the literature, several methods to approximate the original matrix by rank reduction have been proposed. The aim of these methods varies from discovering high-order relations between entries to improving efficiency by reducing its noise and dimensionality. We exploit three methods for building our semantic spaces: Latent Semantic Analysis (LSA), Random Indexing (RI) and LSA over RI.

All these methods produce a new matrix $\hat{\mathbf{M}}$, which is a $n \times k$ approximation of the co-occurrence matrix $\mathbf{M}$ with n row vectors corresponding to vocabulary terms, while k is the number of reduced dimensions.

3.1.2 Latent Semantic Analysis

Latent Semantic Analysis [Deerwester et al., 1990] is based on the Singular Value Decomposition (SVD) of the original matrix $\mathbf{M}$. $\mathbf{M}$ is decomposed in the product of three matrices $\mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top$, where $\mathbf{U}$ and $\mathbf{V}$ are orthonormal matrices

[1] The corpus could be the collection of documents indexed by the QA system, but also some external text collection.

	drink	a	glass	of	wine	beer	is	made	grapes	hops
drink	0	2	2	2	1	1	0	0	0	0
a	2	0	2	2	1	1	0	0	0	0
glass	2	2	0	2	1	1	0	0	0	0
of	2	2	2	0	2	2	2	2	1	1
wine	1	1	1	2	0	0	1	1	1	0
beer	1	1	1	2	0	0	1	1	0	1
is	0	0	0	2	1	1	0	2	1	1
made	0	0	0	2	1	1	2	0	1	1
grapes	0	0	0	1	1	0	1	1	0	0
hops	0	0	0	1	0	1	1	1	0	0

Figure 3.3: An example of $term \times term$ matrix $\mathbf{M}$

whose columns are the *right* and *left eigenvectors* of the matrices $\mathbf{M}^\top\mathbf{M}$ and $\mathbf{M}\mathbf{M}^\top$ respectively, while $\mathbf{\Sigma}$ is the diagonal matrix of the *singular values* of $\mathbf{M}$ placed in decreasing order.

SVD can be applied to any rectangular matrix, and if r is the *rank* of $\mathbf{M}$, then the matrix $\widetilde{\mathbf{M}} = \mathbf{U_k}\mathbf{\Sigma_k}\mathbf{V_k}^\top$ of rank $k \ll r$, built choosing the top k singular values, is the best rank k approximation of $\mathbf{M}$. The approximated $\widetilde{\mathbf{M}}$ s shown in Figure 3.4.

SVD helps both to discover high-order relations between terms and to reduce the sparsity of the original matrix. Moreover, since the matrix $\mathbf{M}\mathbf{M}^\top$ corresponds to all possible combinations of any two terms, it is possible to compute the similarity between two terms by exploiting the relation

$$\mathbf{M}\mathbf{M}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{V}^\top\mathbf{V}\mathbf{\Sigma}^\top\mathbf{U}^\top = \mathbf{U}\mathbf{\Sigma}\mathbf{\Sigma}^\top\mathbf{U}^\top = (\mathbf{U}\mathbf{\Sigma})(\mathbf{U}\mathbf{\Sigma})^\top$$

In the case of the k-approximation of $\mathbf{M}$, the complexity of the computation of the similarity between any two terms is reduced.

3.1.3 Random Indexing

We exploit Random Indexing (RI), introduced by Kanerva [1988], for creating the DSM based on RI. This technique allows us to build a semantic space with no need for matrix factorization, because vectors are inferred using an incremental strategy. Moreover, it allows to solve efficiently the problem of reducing

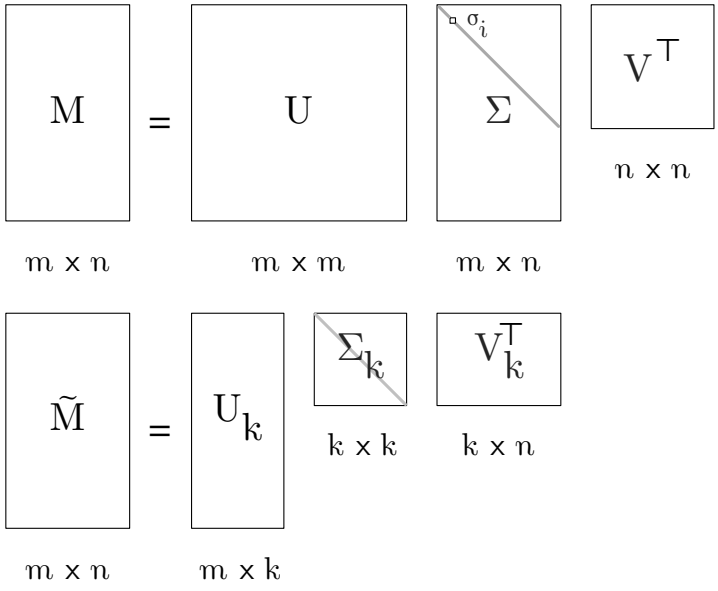


Figure 3.4: A depiction of SVD matrices

dimensions, which is one of the key features used to uncover the "latent semantic dimensions" of a word distribution.

RI is based on the concept of Random Projection according to which randomly chosen high dimensional vectors are "nearly orthogonal" (they are not formally orthogonal, but in practice their inner product results to be 0 or really close to 0). This yields a result that is comparable to orthogonalization methods, such as SVD [Landauer and Dumais, 1997], but saving computational resources.

Formally, given an $n \times m$ matrix $\mathbf{M}$ and an $m \times k$ matrix $\mathbf{R}$ made up of m k-dimensional random vectors, we define a new $n \times k$ matrix $\mathbf{M}'$ as follows:

$$\mathbf{M'}_{n,k} = \mathbf{M}_{n,m}\mathbf{R}_{m,k} \quad k << m \tag{3.1}$$

The new matrix $\mathbf{M}'$ has the property to preserve the distance between points. This property is known as Johnson-Lindenstrauss lemma [Johnson and Lindenstrauss, 1984]: if the distance between any two points of $\mathbf{M}$ is d, then the distance d_r between the corresponding points in $\mathbf{M}'$ will satisfy the property that $d_r = c \cdot d$. A proof of that property is reported in [Dasgupta and Gupta, 1999].

The product between $\mathbf{M}$ and $\mathbf{R}$ is not actually computed, but it corresponds to building $\mathbf{M}'$ incrementally, as follows:

1. Given a corpus, a *random vector* is assigned to each term. The random vector is high-dimensional, sparse and with very few elements with non-

zero values $\{-1, 1\}$, which ensures that the resulting vectors are nearly orthogonal, and the structure of this vector follows the hypothesis behind the concept of Random Projection.

2. The *semantic vector* of a term is given by summing the random vectors of terms co-occurring with the target term in a predetermined context (document / sentence / window).

Having $\{-1, 1\}$ as the only possible values halves the probability of generating random vectors with 1 in the same position, which will result in non orthogonal vectors.

An example of the construction of the term vectors is shown in Figure 3.5

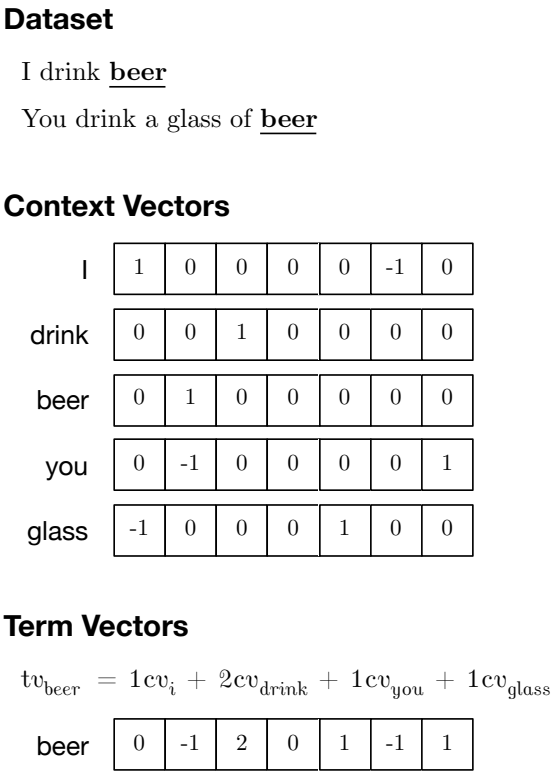


Figure 3.5: Term Vector construction in Random Indexing

3.1.4 Latent Semantic Analysis over Random Indexing

Computing LSA on the co-occurrence matrix $\mathbf{M}$ can be a computationally expensive task, as the vocabulary V can reach thousands of terms. Here we propose a simpler computation based on the application of the SVD factorization to

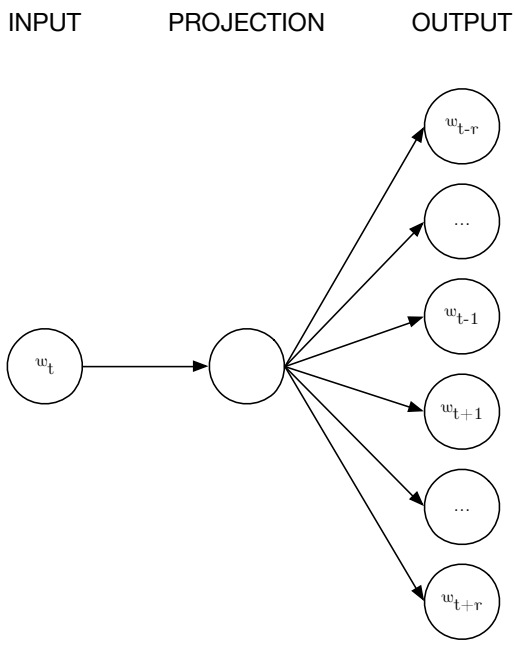


Figure 3.6: The architecture of the Continuous Skip-gram Model

$\mathbf{M}'$, the reduced approximation of $\mathbf{M}$ produced by Random Indexing. Sellberg and Jönsson [2008] followed a similar approach for the retrieval of similar FAQs in a QA system. Their experiments showed that reducing the original matrix by RI resulted in a drastic reduction of LSA computation time. The trade-off to be paid was the slight worse performance, which were better than TTM and RI anyway.

3.1.5 Continuous Skip-gram Model

A quite different DSMs aims at learning distributed representations of words with neural networks, because they have better performances than LSA in preserving linear regularities among words [Mikolov et al., 2013a] and the latest models are computationally less expensive, so they scale better on large data sets.

In [Mikolov et al., 2013b], the authors construct a really scalable log-linear classification network, using a simpler architecture than previous work, where neural networks are usually constructed with several non-linear hidden layers [Bengio et al., 2003]. Two such simpler networks are proposed in that work: the Continuous Bag-of-Words Model and Continuous Skip-gram Model. While both are shown to be effective in semantic-syntactic word relationship learning and sentence completion tasks, the former is faster to train, while the latter has

better performances at the cost of slightly longer training time. Although both are really scalable, for our experiments we decided to adopt the latter one for its accuracy.

The Continuous Skip-gram Model builds on Feedforward Neural Networks described in [Bengio et al., 2003], but it consists only of input, projection and output layers, so removing the hidden layer. As most of the complexity is caused by the non-linear hidden layer, this improves the learning efficiency at the expenses of a representation that might be less precise, but enables to learn models with bigger amounts of data. The model, shown in Figure 3.6, iterates over the words in the dataset and uses each word w_t as an input to a log-linear classifier with continuous projection layer. What it outputs is a prediction of the words within a certain range before and after the input word.

As the words that are more distant from the input word are less related to it, the model adopts a randomization policy: if c is the fixed range before and after a word, a value r is obtained picking randomly a value between $[1, c]$. Then r words before the current and r words after the current are used as correct labels, from w_{t-r} to w_{t-1} and from w_{t+1} to w_{t+r}.

At the end of the training phase, the weights associated with the projection layer are used as vector representations for each word. The resulting encoding captures meaningful word representations, where words of similar meaning have nearby representations.

3.2 Distributional Semantic Models Integration in Question Answering

The idea behind the application of DSMs in a QA framework is to build new features relying on semantic spaces, and add them to the feature extractor pipeline described in Section 2.3.3. We call this component *Distributional Feature Extractor* and it aims at computing the semantic similarity between the question and each candidate answer.

We use word vector representations for building sentence level vector representation by summing the vectors of the words that appear in the sentence. This way we obtain vector representations for questions and answers and we can compute their cosine similarity to obtain a semantic similarity measure. This measure becomes one feature used in the ranking of the answers. Questions and answers are usually short pieces of text and this makes this strategy more suitable.

In DSMs, given the vector representation of two words $\mathbf{u} = (u_1, \ldots, u_n)^\top$ and $\mathbf{v} = (v_1, \ldots, v_n)^\top$, it is always possible to compute their similarity as the cosine of the angle between them:

$$\cos(\mathbf{u}, \mathbf{v}) = \frac{\sum_{i=1}^{n} u_i v_i}{\sqrt{\sum_{i=1}^{n} u_i^2 \sum_{i=1}^{n} v_i^2}} \tag{3.2}$$

However, the user's question and the candidate answer are sentences composed by several terms, so in order to compute the similarity between them we need a method to compose the words occurring in these sentences. It is possible to combine words through vector addition (+). This operator is similar to the superposition defined in connectionist systems [Smolensky, 1990], and corresponds to the point-wise sum of components:

$$\mathbf{s} = \mathbf{u} + \mathbf{v} \tag{3.3}$$

where $s_i = u_i + v_i$.

Addition is a commutative operator, which means that it does not take into account any order or underlying structures existing between words in both questions and answers. We do not exploit more complex methods to combine word vectors as they do not clearly outperform the simple vector addition [Mitchell and Lapata, 2010].

Given a phrase or sentence s, we denote with $\mathbf{s}$ its vector representation obtained applying addition operator (+) to the vector representation of terms it is composed of. Furthermore, it is possible to compute the similarity between two phrases / sentences exploiting the cosine similarity between vectors (3.2).

Formally, if $q = q_1, q_2, \ldots, q_n$ and $a = a_1, a_2, \ldots, a_m$ are the question and the candidate answer respectively and each q_i and a_i is a term present in them, we build two vectors $\mathbf{q}$ and $\mathbf{a}$ which represent respectively the question and the candidate answer in a semantic space. Vector representations for question and answer are built applying the addition operator to the vector representation of words belonging to them:

$$\begin{aligned} \mathbf{q} &= q_1 + q_2 + \ldots + q_n \\ \mathbf{a} &= a_1 + a_2 + \ldots + a_m \end{aligned} \tag{3.4}$$

The similarity between $\mathbf{q}$ and $\mathbf{a}$ is computed as the cosine similarity between them.

For example, we want to compare the question $\mathbf{q}$ *"Is Guinness a kind of beer?"* with the passage $\mathbf{a^1}$ *"Guinness produces different kinds of stouts"* and the passage $\mathbf{a^2}$ *"Apple produces different kinds of computers"*. The vector representations of the (non-stopword) words are:

$v_{\text{is}} = [0.1, 0.2, 0.3, 0.25]$

$v_{\text{guinness}} = [0.7, 0.1, 0.12, 0.09]$

$v_{\text{kind}} = [0.2, 0.1, 0.65, 0.5]$

$$v_{\text{beer}} = [0.8, 0.05, 0.1, 0.12]$$

$$v_{\text{produces}} = [0.3, 0.4, 0.1, 0.04]$$

$$v_{\text{different}} = [0.1, 0.21, 0.1, 0.12]$$

$$v_{\text{kinds}} = [0.22, 0.08, 0.67, 0.48]$$

$$v_{\text{stouts}} = [0.82, 0.04, 0.11, 0.11]$$

$$v_{\text{apple}} = [0.44, 0.71, 0.24, 0.14]$$

$$v_{\text{computers}} = [0.05, 0.84, 0.2, 0.6]$$

It is easy to see how the vectors for *beer* and *stout* and the vectors for *kind* and *kinds* are really similar to each other (i.e. close in the semantic space).

The representation for $\mathbf{q}$, $\mathbf{a^1}$ and $\mathbf{a^2}$ are the following:

$$\mathbf{q} = v_{\text{is}} + v_{\text{guinness}} + v_{\text{kind}} + v_{\text{beer}} = [1.8, 0.45, 1.17, 0.96]$$

$$\begin{aligned}\mathbf{a^1} &= v_{\text{guinness}} + v_{\text{produces}} + v_{\text{different}} + v_{\text{kinds}} + v_{\text{stouts}} \\ &= [2.14, 0.83, 1.1, 0.84]\end{aligned}$$

$$\begin{aligned}\mathbf{a^2} &= v_{\text{apple}} + v_{\text{produces}} + v_{\text{different}} + v_{\text{kinds}} + v_{\text{computers}} \\ &= [1.11, 2.24, 1.31, 1.38]\end{aligned}$$

The cosine similarity among the $\mathbf{q}$ and the two passages $\mathbf{a^1}$ and $\mathbf{a^2}$ is:

$$\cos(\mathbf{q}, \mathbf{a^1}) = 0.9846$$

$$\cos(\mathbf{q}, \mathbf{a^2}) = 0.7794$$

So $\mathbf{a^1}$ would be ranked higher than $\mathbf{a^2}$ in a rank list.

A graphical depiction of word vectors v and their composition in question vector q and answer vector a and the angle θ_{qa} among them is shown in Figure 3.7

The value of the cosine similarity is calculated by the *Distributional Feature Extractor* and used as a feature for ranking. Changing the DSM from which we take the vector representations of the words and changing the corpora employed to learn those representations, we can compute several features. In the description of the experiments in Chapter 5 and Chapter 6 we will describe the exact models and corpora adopted in each case.

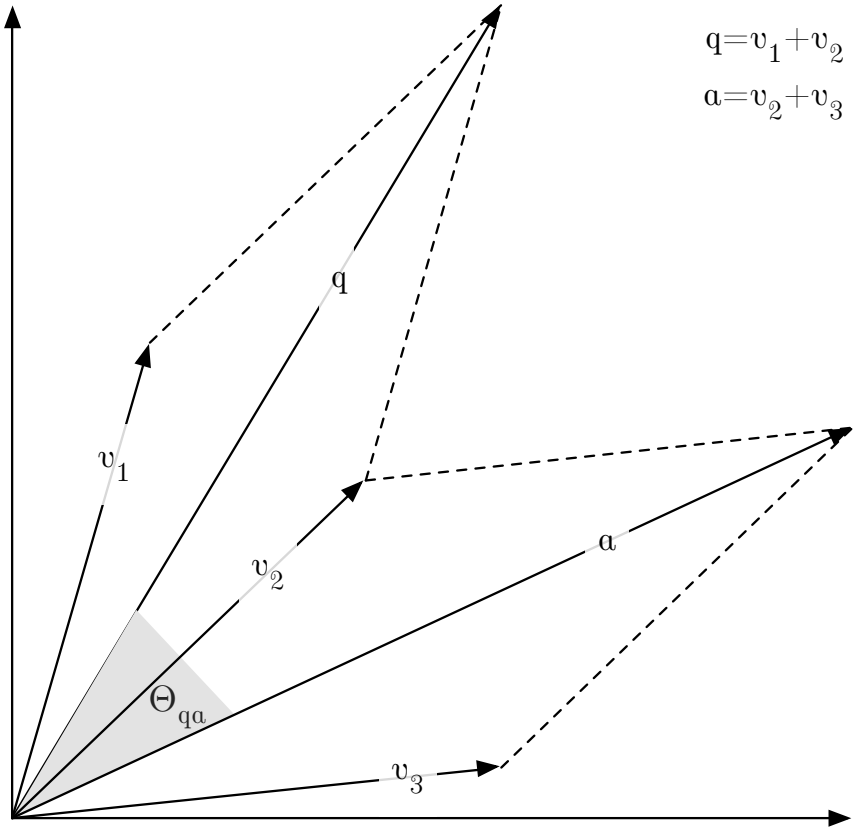


Figure 3.7: Semantic similarity among question vector q and answer vector a

4

Learning to Rank

"We can say that Muad'Dib learned rapidly because his first training was in how to learn. And the first lesson of all was the basic trust that he could learn."

– Frank Herbert, *Dune*

"If history repeats itself, and the unexpected always happens, how incapable must Man be of learning from experience."

– George Bernard Shaw

In this chapter we describe the Learning to Rank setting we adopt for combining different ranking criteria, giving an overview of the main approaches and describing the algorithm we adopt.

4.1 Machine Learning for Ranking

Learning to Rank refers to machine learning techniques for training the model in a ranking task. Ranking is at the core of information retrieval: given a query, candidates documents have to be ranked according to their relevance to the query. Learning to Rank is a relatively new field in which machine learning algorithms are used to learn this ranking function. It is of particular importance for web search engines to accurately tune their ranking functions as it directly affects the search experience of millions of users. A typical setting in Learning to Rank is that feature vectors describing a query-document pair are constructed and relevance judgments of the documents to the query are available. A ranking function is learned based on this training data, and then applied to the test data.

Learning to Rank can be employed in a wide variety of applications in Information Retrieval, Natural Language Processing, and Data Mining. Typical applications are document retrieval, expert search, definition search, collaborative filtering, keyphrase extraction, document summarization, and machine

translation [Li, 2011]. It has been used also for Question Answering [Agarwal et al., 2012]. Intensive studies have been conducted on the problem and significant progress has been made [Liu, 2011] and several machine learning algorithms have been applied to it [Liu, 2009, Freund et al., 2003, Burges et al., 2005, Cao et al., 2006, Xu and Li, 2007, Cao et al., 2007, Cossock and Zhang, 2008, Burges et al., 2011].

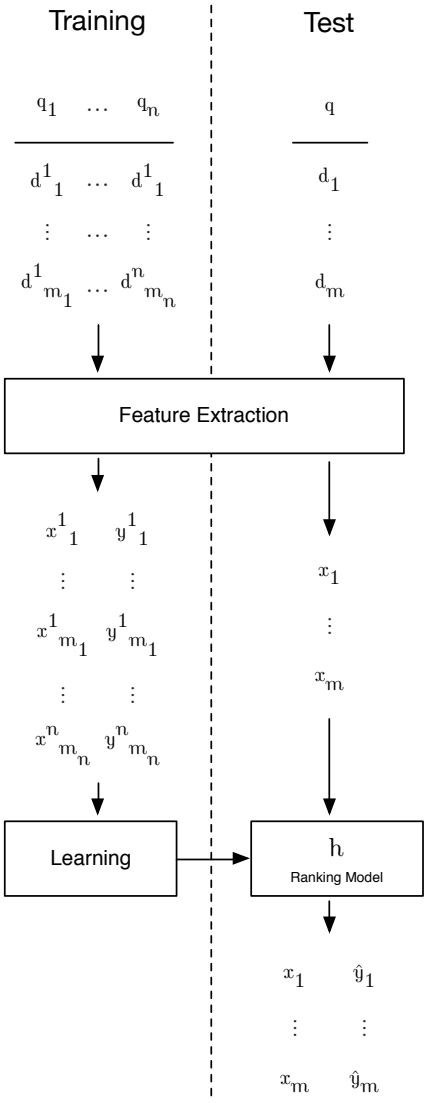


Figure 4.1: Learning to Rank setting

In the Learning to Rank setting (see Figure 4.1), query-document pairs (q, d) are labeled with relevance judgments that indicate the degree of relevance of

the document d with respect to query q. For example, each relevance judgment can be one element in the ordinal set, $perfect, excellent, good, fair, bad$ and is labeled by human editors. The label can also simply be binary: relevant or irrelevant. Each query and each of its documents are paired together, and each query-document pair is represented by a feature vector. The features are usually an indication of the degree of similarity between q and d, but also information about q and d in isolation, such as their length or the PageRank of web documents.

Each pair is treated as a single datapoint and a set of datapoints can be used for training purposes, in order to learn a function to predict the best ranking of different documents according to a query.

Thus the training data can be formally represented as: (x_j^q, y_j^q), where q goes from 1 to n, the number of queries, j goes from 1 to m_q , the number of documents for query q, $x_j^q \in \mathbb{R}^d$ is the d-dimensional feature vector for the pair of query q and the j-th document for this query and y_j^q is the relevance label for x_j^q.

This training data is used for training a model h that is used to output a ranking score $\hat{y}$ for all the feature vectors x obtained from the (q, d) pairs in the test data.

In our experiments the queries are the questions and the documents are the candidate answers.

4.2 Approaches

Several algorithm have been proposed for this goal in the literature [Liu, 2009]. They can be categorized, with few exceptions like query dependent ranking [Geng et al., 2008] and multiple nested ranking [Matveeva et al., 2006], in three major families: Pointwise approach, Pairwise approach and Listwise approach.

The pointwise and pairwise approaches directly cast the problem of learning a ranking function to a classification, regression, or ordinal classification problem. On the other hand, listwise approaches use document ranking lists directly as instances for learning learning and learn a ranking model based on the whole lists. The main differences among the approaches is the loss functions they try to optimize.

4.2.1 Pointwise

In the pointwise approach, the group structure of ranking is ignored and the setting becomes typical supervised learning, with data representing a mapping from x to y. If y is a class label, real number or grade label, then the problem becomes classification, regression, and ordinal classification, respectively and it is possible to use existing methods to perform the learning task.

Given a query, these models output a value that can be directly used to rank documents, sorting them according to the scores given in output by the model. The loss function in learning is pointwise as it is defined on a single feature vector of the query-document pair.

Some pointwise algorithms are Prank [Crammer and Singer, 2001], OC SVM [Shashua and Levin, 2002], McRank [Li et al., 2007], and Subset Ranking [Cossock and Zhang, 2006].

4.2.2 Pairwise

In the pairwise approach, the group structure of ranking is ignored like in the pointwise approach, but the setting is casted to pairwise classification or pairwise regression. In those tasks the classifiers are used to order document pairs with respect to a query, and the same learned function can be used to full document ranklist ordering.

In the pairwise approach, data is taken in pairs of examples with different labels from the ranklist of the same query. From the dataset (x_j^q, y_j^q), if $y_i^q > y_k^q$, meaning y_i^q has an higher grade than y_k^q, then (x_i^q, x_k^q) becomes a preference pair where the first element is the vector is the one with higher grade label y with respect to the second element. The pairs are used as instances in the pairwise classification or regression problem.

Like in pointwise approaches, given a query, these models output a value that is assigned to each document and can be directly used to rank them. An effective model for pairwise classification or regression is also an effective model for ranking. The loss function in learning is pairwise as it is defined on a pair of feature vectors obtained from two query-document pairs.

Some of the most widely adopted pairwise algorithms are RankSVM [Joachims, 2002], RankBoost [Freund et al., 2003], RankNet [Burges et al., 2005], LambdaRank [Burges et al., 2006] and LambdaMART [Wu et al., 2010].

4.2.3 Listwise

The ranking problem, by definition, deals with a set of documents to be ranked against a query, and listwise approaches addresses the problem in a more direct way, using entire ranking lists as instances in both learning and prediction, exploiting the group structure of the ranklist. As most evaluation measures are computed on full ranklists, listwise models can try to optimize for them more directly, incorporating them in the loss functions.

In the listwise approach, data is taken as full ranklists of documents for the same query. From the dataset (x_j^q, y_j^q), for each query q_i, all the vectors $x_j^{q_i}$ with the labels $y_j^{q_i}$ associated with the query are taken together as one instance. The model learns a ranking model that assigns a score to feature

vectors obtained from query-document pairs that can be used for ranking them. Differently from pointwise and pairwise approaches, this is a new problem and classic machine learning algorithms cannot be directly applied, even if the basic guiding principles in the newly developed listwise Learning to Rank algorithms can be mapped back to them.

The listwise approach includes ListNet [Cao et al., 2007], AdaRank [Xu and Li, 2007], SoftRank [Taylor et al., 2008], and AppRank [Qin et al., 2010].

4.3 Evaluation Metrics

The most used metrics for evaluating a Learning-to-Rank algorithm are *Mean Reciprocal Rank*, used when the relevance levels are binary, and *Normalized Discounted Cumulative Gain*. We will use these metrics in our experiments, alongside with other metrics specifically tailored for the experimental setting.

4.3.1 Mean Reciprocal Rank

The *Reciprocal Rank* of a query response is the multiplicative inverse of the rank of the first correct answer. It varies from 0 to 1, with a score of 1 meaning that the first document is the correct answer and 0 meaning that no correct answer was found in the retrieved document list. The *Mean Reciprocal Rank (MRR)* is the average of the reciprocal ranks of results for a set of queries Q, calculated as follows:

$$\mathrm{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\mathrm{rank}_i}$$

where $|Q|$ is the number of queries and rank_i is the rank of the first correct answer.

4.3.2 Normalized Discounted Cumulative Gain

The *Normalized Discounted Cumulative Gain (NDCG)* measures the performance of a ranking algorithm based on the graded relevance of the documents. It varies from 0 to 1, with 1 representing the ideal ranking of the entities. This metric is commonly used in Information Retrieval and to evaluate the performance of web search engines.

The premise of the *Discounted Cumulative Gain* is that highly relevant documents appearing lower in a search result list should be penalized as the graded relevance value is reduced logarithmically proportional to the position of the result. It is defined as follows:

$$\mathrm{DCG_k} = \sum_{i=1}^{k}\frac{2^{rel_k}-1}{\log_2(i+1)}$$

where k is a specified position in the ranking and rel_k is an indicator function of relevance that returns 1 if the answer in the k^{th} position in the ranking is the best answer. Calculating $IDCG_k$ as the maximum possible (ideal) DCG for a given set of queries, documents and relevances, the DCG can be normalized to map the possible scores to 0 to 1 range, obtaining the $nDCG$:

$$nDCG_k = \frac{DCG_k}{IDCG_k}$$

4.4 Random Forests

We wanted to adopt a Learning to Rank algorithm to combine all the features available for extraction (see Section 2.3.3 and Section 6.3) with the features based on DSMs presented in Chapter 3 and with several other features that depend on the specific experiments (see Section 6.3).

We opted for Random Forests (RF) [Breiman, 2001] because of their resilience to overfitting, a problem that may affect our experimental setting due to the size of our dataset, and because of the successful results in several use cases related to Community-based Question Answering [Dalip et al., 2013] and in other large scale retrieval experiments [Mohan et al., 2011] where, in general, decision trees were the most popular class of models among the top competitors and ensemble methods, including boosting, bagging and RF, were dominant techniques.

Furthermore, in our QA settings, the relevance judgments are always binary, and the distribution of positive and negative examples is skewed towards the negative examples, with usually only one positive example for each query. This is likely to make more complex approaches like listwise ones (that usually achieve better performances in IR benchmarks) less useful because most of the items in the ranking will have the same relevance judgment and their ordering is not relevant. Our experiments confirmed empirically this intuition (see Section 6.6.5).

Let $x^j = \phi(q, d)$, where ϕ is a feature extractor, and x^j is a m-dimensional vector. Let $D = (x^1, y^1), \ldots, (x^n, y^n)$ be a set of feature vectors extracted from query-document pairs $x^i \in \mathbb{R}^d$ and their associated relevance ratings $y^i \in Y$.

The RF algorithm trains a model H such that $H(x^j) \approx y^j$ and so that the ranking of all the documents d appearing in pair with a query q according to $H(x^j)$ is as similar as possible to the ranking according to y^j.

The algorithm is shown in Algorithm 1.

The main idea of RF is to apply a decision tree regression algorithm to M subsets of D and then average the results. A sample D_t is extracted with replacement from D (step 2). A set K of features is randomly picked from the feature set, so that $|K| \leq m$ (step 3). Learning a tree, if one or a few features are very strong predictors for the response variable (target output), these features

Algorithm 1 Random Forests

Require: $D = (x^1, y^1), \ldots, (x^n, y^n), r > 0$

1: **for** $i \leftarrow 1$ to r **do**
2: $\quad D_t \leftarrow sample(D)$
3: $\quad K \leftarrow randomPick(m)$
4: $\quad h_i \leftarrow buildDecisionTree(D_t, K)$
5: **end for**
6: $H() \leftarrow \frac{1}{r} \sum_{i=1}^{r} h_i()$
7: **return** $H()$

will be selected in many of the r trees, causing them to become correlated. To contrast this, for a dataset with m features, $\sqrt{m}$ features are used in each split. A decision tree (CART) is induced from D_t using the features in K (step 4). The whole process is repeated r times and the outputs of all the single trees are averaged to obtain the function H (step 6). The use of different samples of the data from the same distribution and of different sets of features for learning the individual decision trees prevent overfitting.

In our experimental evaluation we adopted the implementation provided by the RankLib library[1] with the default parameters.

[1] `http://sourceforge.net/p/lemur/wiki/RankLib/`

Part II

Experiments

5

Passage Retrieval Experiments

"It is not that I'm so smart. But I stay with the questions much longer."
– Albert Einstein

"There are no answers, only cross references."
– Norbert Wiener, *Wiener's Law of Libraries*,

In this chapter we describe a preliminary experiment carried out on the *ResPubliQA 2010 Dataset*. The aim is to test the effectiveness of our proposed method of integration of DSMs in a QA system.

With this experiment we aim at answering the first research questions, RQ1:

- *RQ1. Is the Distributional Semantics representation a good representation for the meaning of questions and answers?*
  We address this question evaluating the effectiveness of the ranking of answers obtained with the Distributional Semantics representations.

5.1 Experiment Description

The goal of the evaluation is twofold: (1) assess the effectiveness of DSMs into our QA system and (2) provide a comparison between the several DSMs adopted by the *Distributional Feature Extractor*.

The task we choose to perform in order to answer the goals is passage retrieval: given a set of natural language questions q, the goal is to find the passage of text that contains the correct answer in a specific collection of documents (already split in paragraphs). There could be more than one passage of text that contains the correct answer and there is no candidate passage to each question,

all passages from all documents in the collection could potentially contain the correct answer.

In order to test the effectiveness of the DSMs for QA, we rely on the QA framework presented in Chapter 2.

In particular, we adopted four types of semantic spaces: a classical Term-Term co-occurrence Matrix (TTM) used as baseline, Latent Semantic Analysis (LSA) applied to TTM, Random Indexing (RI) approach to reduce TTM dimensions, and finally the LSA over RI (LSARI). All the DSMs are constructed on the corpus of documents used for the evaluation. The *Distributional Feature Extractor* will assign a score based on the similarity between the question and the candidate answers inside the DSMs.

5.2 Dataset

The evaluation has been performed on the *ResPubliQA 2010 Dataset* adopted in the *2010 CLEF QA Competition* [Peñas et al., 2010]. Two sets of documents have been included in *ResPubliQA 2010* collection: a subset of the *JRC-ACQUIS Multilingual Parallel Corpus* and a small portion of the *EUROPARL* collection. Both are multilingual parallel collections.

JRC-ACQUIS is a freely available parallel corpus containing the total body of European Union documents, mostly of legal nature. It comprises contents, principles and political objectives of the European Union treaties, the European Union legislation, declarations and resolutions, international agreements and acts. Texts cover various subject domains, including economy, health, information technology, law, agriculture, food, politics and more. This collection of legislative documents currently includes selected texts, written between 1950 and 2006, with parallel translations in 22 languages. The corpus is encoded in XML, according to the TEI guidelines.

The subset used in *ResPubliQA* consists of 10,700 parallel and aligned documents per language (Bulgarian, English, French, German, Italian, Portuguese, Romanian and Spanish). The documents are grouped by language, and inside each language directory, documents are grouped by year. All documents have a numerical identifier called the CELEX code, which helps to find the same text in the various languages. Each document contains a header (giving for instance the download URL and the EUROVOC codes) and a text (which consists of a title and a series of paragraphs).

EUROPARL is a collection of the *Proceedings of the European Parliament* dating back to 1996. European legislation is a topic of great relevance to many potential users from citizens to lawyers, government agencies, politicians and many others. *EUROPARL* comprises texts in each of the 11 official languages of the European Union (Danish, German, Greek, English, Spanish, Finnish,

French, Italian, Dutch, Portuguese and Swedish). With the enlargement of the European Union to 25 member countries in May 2004, the European Union has begun to translate texts into even more languages. However, translations into Bulgarian and Romanian start from January 2009 and for this reason we only compiled documents from the European Parliament site[1] starting from that date. In this way, we ensured a parallel collection for 9 languages (Bulgarian, Dutch, English, French, German, Italian, Portuguese, Romanian and Spanish).

For our experiments we adopted only the English and Italian parts of the dataset. The questions to answer are 200 and the relevance of the passages of text with respect to the question has been manually assessed, with only one correct passage for each question is present in the gold standard.

Moreover, the results submitted by the participants in the competition have been manually evaluated and few more passages have been added to the gold standard. This resulted in a slightly expanded gold standard, with an average of 2.2 correct answers for each question and each system in the competition received on average a 11% improvement from the expanded gold standard. Furthermore, it is not possible to directly compare with the results of the competition, as some answers provided by our system could be correct, but there is no human assessor that can confirm that. The dataset and the extended gold standard are anyway a useful tool and we provide our own baselines to compare against.

5.3 Analysis of the 2010 CLEF QA Competition Participants

The groups participating in the *2010 CLEF QA Competition* were 13. An analysis of the most effective systems shows some important features of the systems: Toba et al. [2010] used lemmas for indexing the passages while Nemeskey [2010] and Sabnani and Majumder [2010] used stems and terms. Few other systems indexed n-grams and chunks, only one indexed word senses. All three systems performed a question classification step, relying mostly on manual patterns, while Nemeskey [2010] also adopted lemmatization and PoS tagging for this subtask. The retrieval was usually carried out using the BM25 model, which was used also as a baseline for the passage retrieval task.

Different ranking and validation techniques were applied by participants, the most common processing was to measure the lexical overlapping among questions and candidate answers (it was performed by 5 participants). On the other hand, more complex techniques such as syntactic similarity or theorem proving were applied by very few participants. Few others adopted redundancy

[1] http://www.europarl.europa.eu/

based measures and only one relied on a learned combination of criteria.

For those systems performing also answer selection, the most common processing was the use of named entities, numeric and temporal expressions, while some systems relied on syntactic processing by means of chunking, dependency parsing or syntactic transformations. Only one relied on a logic representation with a theorem prover.

5.4 Experimental Setup

The metric adopted in the experiment is the accuracy $A@n$, calculated considering only the first n answers. If the correct answer occurs in the top n retrieved answers, the question is marked as correctly answered. In particular, we take into account several values of n =1, 5, 10 and 30. This metric is sometimes also referred to as $CORRECT@n$ and usually is only taken into account with n =1, making $A@1$ equal to $Precision@1$. We use this metric with different values for n as we would want to analyze the performance of our system considering acceptable for a potential user to find the correct answer not only first in the rank, but also in the first n positions. Moreover, we adopt another metric, the **MRR**, that considers the rank of the first correct answer. We already described this metric in Section 4.3.1. In this metric the score is higher if the first correct passage occurs on higher in the ranking. Again, we use this metric because we want to assess the performance of the system in returning the correct answer as high as possible in the raning. Both measures are widely used in IR and QA literature [Moldovan et al., 2003, Forner et al., 2010].

The different DSMs and the classic TTM have been used in two ways:

- as feature extractor alone, which means no other feature extractors are adopted in the pipeline and the only score used for ranking is the one coming from the *Distributional Feature Extractor*,
- combined with some other linguistic similarity features: term overlap (TO), lemma+pos overlap (LO), lemma+pos density (LD) and the exact sequence (E). As the number of questions is rather small (only 200) we preferred not to learn a ranking model and to rely on the CombSum aggregation strategy that we described in Section 2.3.4.

Moreover, we need to setup some parameters of DSMs. The window w of terms considered for computing the co-occurrence matrix is 4, while the number of reduced dimensions considered in LSA, RI and LSARI is equal to 1,000.

The performance of the combination of the linguistic similarity features, without the distributional ones, is shown as a baseline. The experiments have been carried out both for English and Italian, results are shown respectively in Table 5.1 and 5.2. Each Table reports the accuracy $A@n$ computed considering

Table 5.1: Evaluation Results on *ResPubliQA 2010 Dataset* for English.

	Features	A@1	A@5	A@10	A@30	MRR
alone	TTM	0.060	0.145	0.215	0.345	0.107
	RI	0.180	0.370	0.425	0.535	0.267‡
	LSA	**0.205**	**0.415**	**0.490**	0.600	**0.300**‡
	LSARI	0.190	0.405	**0.490**	**0.620**	0.295‡
combined	*baseline*	*0.445*	*0.635*	*0.690*	*0.780*	*0.549*
	TTM	0.535	0.715	0.775	0.810	0.614†
	RI	0.550	**0.730**	0.785	**0.870**	**0.637**†‡
	LSA	**0.560**	0.725	**0.790**	0.855	**0.637**†
	LSARI	0.555	**0.730**	**0.790**	**0.870**	0.634†

Features for combined evaluation: TO+LO+LD+E. Significance of the difference with $p < 0.05$ with respect to the baseline (†) and with TTM (‡) are shown.

a different number of answers, the MRR and the significance of the results with respect to both the baseline (†) and the distributional model based on TTM (‡). The significance is computed using the non-parametric Randomization test, as suggested in [Smucker et al., 2007], since it has proven to be an effective test under several circumstances. For the Randomization test a Perl script supplied by the authors[2] has been employed.

5.5 Experimental Results

Considering each distributional feature on its own, the results show that all the proposed DSMs perform better than the TTM, and the improvement is always significant. The best improvement in English is obtained by LSA (+180%), while in Italian by LSARI (+161%).

Taking into account the distributional features combined with the linguistic similarity ones, the results show that all the combinations are able to overcome the baseline. In English we obtain an improvement, about 16% with respect to the baseline. For the Italian language, we achieve a higher improvement. The best result is obtained by LSARI with an improvement of 26% with respect to the baseline.

The slight difference in performance between LSA and LSARI suggests that LSA applied to the matrix obtained by RI can produce the same results of the LSA applied to TTM, but requiring less computation time, as the matrix obtained by RI contains less dimensions than the original TTM matrix.

[2] http://www.mansci.uwaterloo.ca/~msmucker/software/paired-randomization-test-v2.pl

Table 5.2: Evaluation Results on *ResPubliQA 2010 Dataset* for Italian.

	Features	**A@1**	**A@5**	**A@10**	**A@30**	**MRR**
alone	TTM	0.060	0.140	0.175	0.280	0.097
	RI	0.175	0.305	0.385	0.465	0.241‡
	LSA	0.155	0.315	0.390	0.480	0.229‡
	LSARI	**0.180**	**0.335**	**0.400**	**0.500**	**0.254**‡
combined	*baseline*	*0.365*	*0.530*	*0.630*	*0.715*	*0.441*
	TTM	0.405	0.565	0.645	0.740	0.539†
	RI	0.465	**0.645**	**0.720**	**0.785**	0.555†
	LSA	0.470	**0.645**	0.690	**0.785**	0.551†
	LSARI	**0.480**	0.635	0.690	**0.785**	**0.557**†‡

Features for combined evaluation: TO+LO+LD+E. Significance of the difference with $p < 0.05$ with respect to the baseline (†) and with TTM (‡) are shown.

In general, the results are encouraging and suggest the effectiveness of using DSMs in our QA framework. Moreover, the proposed semantic spaces are able to outperform the classical TTM in our evaluation.

Finally, the improvement obtained considering each distributional feature on its own shows an higher increment than their combinations with the linguistic similarity features. This suggests that a more complex method to combine features should be used in order to strengthen the contribution of each of them. To this purpose, we investigated some learned models for Learning to Rank in the following experiments in Chapter 6.

The results show the effectiveness of the proposed approach, answering RQ1, and highlight that more sophisticated techniques such as LSA and RI are able to outperform the simple term-term matrix used as baseline.

6

Experiments with Community Question Answering

"To exhibit the perfect uselessness of knowing the answer to the wrong question."

– Ursula K. Le Guin, *The Domestication of Hunch*

In this chapter we describe an extended experiment carried out on two large-scale real-world datasets extracted from Yahoo! Answers. The experimental task is best answer prediction, slightly different from the passage retrieval one of the experiment in Chapter 5, having already candidate answers for each question, but the ranking of the answers takes place in the same way. In this experiment we adopt a wide range of features, including the distributional semantic ones, and compare with strong state-of-the-art baselines. We furthermore provide background for Community Question Answering and Expert Finding, as some features we adopt originated in those fields. We provide details about the adopted features and discuss the results of the experiment.

With this experiment we aim at answering two research questions, RQ1 and RQ2:

- *RQ1. Is the Distributional Semantics representation a good representation for the meaning of questions and answers?*
  We address this question evaluating the effectiveness of the ranking of answers obtained with the Distributional Semantics representations.
- *RQ2. Can distributional semantic representations be combined with other criteria in order to obtain a better ranking of the answers?*

To answer this question, we propose to combine Distributional Semantics with several other criteria for answer ranking in a machine learning setting. We experimented on large-scale dataset the combination of criteria and the contribution of each criterion family to the quality of the ranking.

6.1 Experiment Description

Community Question Answering (CQA) sites such as Yahoo! Answers, StackOverflow, or Ask.com have been very popular since the beginning of the rise of the Social Web due to their ability to leverage a collaborative paradigm to serve articulate user queries that may not find satisfactory answer if submitted to conventional search engines [Morris et al., 2010]. Even though the management of the question-answering process in major online services is usually conducted by human users with very little support from automated techniques, in the last years much research has been done to provide automated methods to identify topical experts, automatically routing to them questions relevant to their expertise and select the best answers among the ones provided (see Section 6.2 for an extensive overview). In particular regarding the automatic best answer selection, which is a fundamental task in this field, an impressive amount of work has been conducted, but with a general lack of coherence in the experimental approaches. It is indeed difficult to establish how different methods or feature sets perform in relation to each others and in which cases.

In the general scenario of best answer prediction, there are some issues that we contribute to address with this experiment. First, there is a general lack of understanding of the relative predictive power of different textual and linguistic features that have been analyzed in isolation in several previous papers but have never been directly compared in an extensive, systematic way. Also, the space of linguistic features that are relevant to this task has still several unexplored areas. Second, the interplay between textual approaches and expert-finding methods based on the networks of user interaction have been mainly considered independently and it is not clear how much the overall predictive potential can benefit from their integration. Last, the performance of the majority of best answer prediction methods in the literature has not been tested in relation to different question types or categories.

We contribute in this direction by considering a vast amount of lexical and language features, including semantic role labeling, machine translation and especially a novel class of Distributional-Semantics-based features. We combine them with a set of network-based features for expert finding, that have always been considered in isolation, in a learning to rank approach. Our results have a two-fold implication. First, they allow us to shed light on the relative importance of features that have not been systematically compared in the past. Last, they

led us to produce a supervised model with a predictive accuracy that tops three of the latest, yet already widely popular methods for best answer prediction.

More specifically, in this experiment we make the main following contributions.

- We propose, for the first time a combination of textual features with expertise network features in a learning to rank framework for the task of best answer prediction.
- We consider 225 features in the CQA domain and we test them on the largest dataset from Yahoo! Answers that has been ever used for this task. Some of those feature groups have been already considered in the literature but never compared directly to one another. The most effective combination of features we use reaches up to 27% performance gain in best answer prediction over recent state-of-the-art methods.
- We break down the results by question type, finding that Text Quality features are more suited to predict the best answer to factual and subjective questions, whereas features from the user profile are more predictive for discussion and poll-type questions.
- We introduce new Distributional-Semantics-based features. We verify that the information they convey is not completely overlapping with other well-established textual-based features, thus bringing a considerable additional power to prediction models. We find that those features can replace more costly and widely used textual similarity features without losing predictive performance in the general case.

6.2 Background

In this section we discuss some background work in the field, describing studies on Community Question Answering and Expert Finding, as well as some of the features that are most widely adopted in previous approaches.

6.2.1 Community and non-Factoid Question Answering

Several approaches have been developed for finding and ranking answers in CQA. A particular branch of research focused on Non-Factoid QA systems trying to address the problem of finding answers to *Why* and *How* questions. They often use CQA datasets for evaluations and adopt similar architectures to the CQA answer ranking engines, although focusing more on linguistic features.

One of the earliest approaches to the problem adopted different measures of Text Quality to find the best answer for a given question [Agichtein et al., 2008].

Intrinsic answer qualities such as grammatical, syntactic and semantic complexity, punctuation and typo errors are adopted, along with question-answer similarity and user expertise estimations. We build on that work by picking all the features reported as most effective, expanding them with new categories of features, and using a more robust learning algorithm.

The importance of linguistic features for Non-Factoid QA have been confirmed in several studies [Verberne et al., 2008, 2010, 2011], in which it is shown how the adoption of semantic role labeling based features [Bilotti et al., 2010] and deep and shallow syntactical structures [Severyn and Moschitti, 2012] can improve the performance of a Non-Factoid QA system. We also adopt distributional linguistic features adding even more levels of lexicalization to the linguistic representation.

Another set of approaches adopts machine translation models to learn how to reformulate a question into an answer so that the probability of the translation of question into the answer can be calculated and the candidate answers can be ranked accordingly [Echihabi and Marcu, 2003, Riezler et al., 2007, Berger et al., 2000]. Recently, Matrix Factorization algorithms have been adopted for the same goal [Zhou et al., 2013]. We adopt machine translation features, learning different translation models for different linguistic representations.

The study in the field dealing with the largest-scale dataset has been proposed by Surdeanu et al. [2011]. They adopted a large amount of features, bringing together linguistic features, those based on translation and classical frequency and density ones. They tested their ranking model on a subset of Yahoo! Answers showing the effectiveness of each feature subset. As illustrated in Section 6.4, we compare our method to theirs on the same dataset (*Yahoo! Answers Manner Questions*), adding Distributional Semantics, Text Quality, Expertise Network, and User-based features that were not previously considered.

In more recent years, new approaches based on lexical semantics emerged. Solutions leveraging Wikipedia entities [Zhou et al., 2013] have been also used, showing potential in addressing the retrieval of synonyms and hypernyms. Recurrent Neural Network Language Models [Yih et al., 2013] have been studied as well, confirming that lexical semantics is suitable to tackle the problem.

6.2.2 Expert Finding

A consistent branch of the studies on expert finding consists in casting the problem into an information retrieval problem, using methods to model the relevance of candidate users to a given question or topic. In *profile-based* methods, candidates are described by a textual profile and profiles ranked with respect to an expertise query [Liu et al., 2005, Craswell et al., 2001], while in *document-based* approaches documents relevant to the query are retrieved first, and then can-

didates are ranked based on the co-occurrence of topic and candidate mentions in the retrieved documents [Balog et al., 2006, Serdyukov and Hiemstra, 2008].

Several slight variants to such approaches have been experimented during the few past years, including topic-specific information retrieval approaches, where users' expertise is calculated only from the portion of their past history that is relevant to the question [Li et al., 2011]. The use of topic modeling [Riahi et al., 2012], as well as classification approaches Zhou et al. [2012] as opposed to information retrieval have been explored as well. Most often, these approaches rely on features of a single type or on quite sparse sets of features of multiple types.

It is also worth mentioning that, in some cases, the task of expert finding has been addressed from a slightly different perspective that goes under the label of "question recommendation", that aims to recommend interesting questions for a contributor that is willing to provide answers. Such approaches tend to privilege the perspective of the answerer, for instance trying to assign questions also to people who have never answered, to guarantee higher fairness of the system Kabutoya et al. [2010]. One of the most complete pieces of work in this direction Dror et al. [2011] uses a combination of collaborative filtering and content-based approaches, showing that the content signal is the most powerful to predict good user-question associations.

Alternative solutions to text-based methods mine the network of interactions between users to infer which ones are the most "expert" with respect to a specific question. This task, also known as *expert finding*, is not only limited to CQA portals but it is generally directed to retrieving a ranked list of people who are knowledgeable on a given topic.

This task can be addressed using techniques that are similar to the ones outlined above, such as information retrieval techniques, probabilistic frameworks and topic models Liu et al. [2010]. However, graph-based models are particularly suited to capture the expertise of individual contributors as they interact with their peers.

In any social domain, the expertise may emerge from the complex interactions of users, and can be modeled with the so-called *Expertise Networks* [Zhang et al., 2007a,b], whose construction and structure is domain-dependent and can potentially mix heterogeneous graphs [Smirnova and Balog, 2011, Bozzon et al., 2013]. Examples of Expertise Networks include scientific collaboration networks [Lappas et al., 2009], social networks [Zhang et al., 2007b,b, Bozzon et al., 2013], communication networks [Dom et al., 2003, Fu et al., 2007], folksonomies [Noll et al., 2009], and so on. Specifically, in CQA, as we will detail in Section 6.3.3, the Expertise Networks have been modeled based on the asker-replier information [Jurczyk and Agichtein, 2007], the assignment of the best answer [Bouguessa et al., 2008, Gyongyi et al., 2007], and the competition between answerers [Liu

et al., 2011, Aslay et al., 2013]. In CQA, once the experts in specific domains are identified, algorithms of *question routing* can be used to deliver relevant questions to them, also taking into account their availability [Li and King, 2010, Horowitz and Kamvar, 2010] and workload balance among the group of experts [Chang and Pal, 2013].

Properties of Expertise Networks such as their shape, connectivity, and associativity patterns have been investigated in depth in previous work [Zhang et al., 2007a, Smirnova and Balog, 2011, Jurczyk and Agichtein, 2007, Chen et al., 2006]. In CQA specifically, studies on Expertise Networks include the analysis of user behavior in terms of topical focus and discussion triggering [Gyongyi et al., 2007], the characterization of the type of topics discussed [Adamic et al., 2008], and the relation of tie strength with the effectiveness of the given answers [Panovich et al., 2012].

However, previous literature in CQA has focused mostly on how network of expertise could be leveraged to find the most expert users, as experts can likely provide high-quality answers. The common assumption is that graph *centrality* on expertise network is correlated with expertise, and this has indeed been shown extensively in the context of CQA [Jurczyk and Agichtein, 2007, Aslay et al., 2013]. Standard centrality metrics, such as PageRank and HITS, as well as custom scores like ExpertiseRank [Zhang et al., 2007b] are commonly used for this purpose. Although in the past centrality metrics in CQA Expertise Networks have been found to be less effective in the task of best answer prediction compared to simple baselines such as the personal best answer count or ratio or best answer ratio [Bouguessa et al., 2008, Chen and Nayak, 2008], recent work has shown that some combinations of expertise network and centrality metrics can indeed beat also the best answer ratio, especially for some categories of questions [Aslay et al., 2013].

In network-based frameworks, expertise can be interpreted as topic independent, similarly to the notion of authority on a graph, but expertise in CQA is more often topic-dependent. To address that, a possible solution is to narrow down the focus on topic-induced subgraphs of the whole expertise network, assuming that all the users who participate in it are relevant to the topic [Aslay et al., 2013, Campbell et al., 2003]. Alternatively, hybrid text-network approaches can be used, either with linear combinations of scores modeling subject relevance and user expertise [Kao et al., 2010], or by recurring to topic modeling to measure the relevance of the past users reply history to a specific topic, and link analysis to estimate their authority within that topic [Zhu et al., 2011]. We tackle this problem by accounting topic relevance with textual features and expertise with network features, combining them in a learning to rank fashion.

Last, we point out that, although we focus on centrality-based expert finding, alternative network-oriented approaches have also been explored, such as label-

propagation or random walk algorithms [Fu et al., 2007, Serdyukov et al., 2008] or supervised approaches [Bian et al., 2009, Chen et al., 2012].

6.2.3 Comprehensive Approaches

Very few studies considered combinations of different types of features. The idea of using user interactions, network-based features and quality estimators together for ranking the answers was introduced by Bian et al. [2008]. More recently the same approach was re-proposed, with more features and a more robust learning to rank algorithm, over StackOverflow data [Dalip et al., 2013], focusing on features specifically designed for that dataset, like code blocks analysis. Our approach follows the path of mixing features coming from different fields and adopts the same learning to rank algorithm, but at the same time we introduce several new features, including deeper linguistic ones and expertise based, dropping the ones that are too dataset-specific to preserve generality and we evaluate our approach on a larger scale dataset.

6.3 Adopted Features

Next, we describe the five main families of features that can be extracted from most of CQA sites.

6.3.1 Textual Features

The textual features rely on the assumption that the similarity between the question and the answer and the intrinsic quality of the answer's text are good proxies for the quality of the answer itself. We divide the features in three different families: *Text Quality*, *Linguistic Similarity* and *Distributional Semantics*. The first contains features already used in literature, the second mainly comes from Non-Factoid QA literature, but exploiting new levels of lexicalization, while the third is completely novel and contains features obtained with distributional models built on term co-occurrence.

Text Quality

Text Quality (tq) features aim to estimate the intrinsic quality of an answer by capturing objective properties of the text composition. A summary follows.

Visual Properties This group of features simply evaluate numerically some properties of the text. The features belonging to this group count the number of whitespace violations and the whitespace density in the text of the answer. The same counts oduced for capital letters and capitalization violations, punctuation density and violations, the URLs in the text, the quoted parts of the answer

and so on. The number of capitalized words and the total count of punctuation marks are also counted, for a total of 23 features that are widely adopted in the literature [Agichtein et al., 2008, Dalip et al., 2013]. The features are shown in Table 6.1.

Group tq	**Subgroup** visual property
Count of auxiliary verb	
Count of pronouns	
Count of conjunctions	
Count of prepositions	
Count of occurrences of the verb “to be”	
Count of punctuation marks	
Minimum length of quoted text	
Average length of quoted text	
Maximum length of quoted text	
Number of quotes	
Number of sentences	
Number of capitalized words	
Number of characters	
Number of whitespace violations (lack or redundancy)	
Number of URLs	
Number of words	
Number of capitalization violations (i.e. no capital letter after sentence mark)	
Number of question marks	
Number of punctuation violations (lack or redundancy)	
Number of whitespaces	
Punctuation characters over all characters	
Whitespace characters over all characters	
Capital letters characters over all characters	

Table 6.1: Visual Property features

Readability These features evaluate how easy is to read an answer. They consider the average word length in terms of number of characters and syllables and the ratio of complex words in the answer. They also include commonly used readability indices such as Kincaid, Ari, Coleman-Liau, Flesch, Fog, Lix and Smog, for a total of 16 features that have been already tested in previous work on CQA [Agichtein et al., 2008, Dalip et al., 2013]. The readability indices are modeled to capture the education degree or the number of years of study necessary to understand a text. In practice, they all combine heuristically quantitative metrics such as the average length of the sentences and average length of the words, the number of characters and syllables, counts of single-syllable

and multi-syllable words, the presence of the words in a whitelist. In particular, the Automated Readability Index is used for both answers and question. The main motivation is that it is most suited for shorter texts like questions, as it is adjusted for number of sentences, and it does not consider syllables but just words and characters, so it is easier and faster to compute.

The features are shown in Table 6.2.

Group tq	**Subgroup** readability
Average words per sentence	
Average words length in syllables	
Average words length in characters	
Number of complex words over all words	
Number of unique words	
Average unique words per sentence	
Flesch-Kinkaid Grade Level	
Automated Readability Index	
Coleman-Liau Index	
Flesch Reading Ease	
Gunning-Fog Index	
LIX score	
SMOG grade	
Number of short sentences	
Number of long sentences	
Automated Readability Index of the question	

Table 6.2: Readability features

Informativeness This group of features was considered since a reasonable answer must contain some information that is not in the question, so we adopt 3 simple features that count the amount of nouns, verbs and adjectives occurring in the answer but not in the question. The features are shown in Table 6.3.

Group tq	**Subgroup** informativeness
Number of nouns present in the answer but not in the question	
Number of verbs present in the answer but not in the question	
Number of adjectives present in the answer but not in the question	

Table 6.3: Informativeness features

Linguistic Similarity

To the best of our knowledge, the most complete approach for generation of Linguistic Similarity (ls) features has been considered by Surdeanu et al. [2011].

They adopt different levels of linguistic representation of a text that can be obtained using NLP algorithms to construct tokens that are then given in input by different similarity and overlap measures. This part of our work follows the same approach.

Having analyzed both questions and candidate answers with an NLP pipeline allows us to build representations of the text using different lexicalization levels: words, stems, lemmas, lemma and PoS tag concatenations, named entities and super senses as tokens. The representations are lists of token n-grams. As an example, the sentence *"The man plays the piano"*, after stopword removal, can be represented as word unigrams (*man, plays, piano*) or as lemma+pos unigrams (*man:NN, play:VBZ, piano:NN*) or as super-sense bigrams (*noun.person-verb.competition, verb.competition-noun.artifact*).

We also tag the text with dependency parsing and semantic role labeling [Gildea and Jurafsky, 2002], so we can extract chains from them in the same way we extract the n-grams. For the dependency parsing the chains are constructed in the form of *"dependant-relationType-head"* but we can extract also more general chains that do not contain the relationType. For the semantic role labeling, the chain has the form of *"predicate - argumentType - argument"*. Also in this case the argument type can be omitted. The length of the chain can be increased concatenating the chains of length one that share intermediate elements. For example, by concatenating unlabeled dependencies from the previous example we obtain the chains (*"man - plays", "piano - plays"*).

Because longer chains do not usually add valuable information because of their sparsity [Surdeanu et al., 2011] we decide not adopt them. The tokens that compose the chain can also be at different lexicalization degrees, but to minimize the sparsity we adopted only lemmas and super senses. As for our example, from the sentence *"The man plays the piano"* we extract labeled dependencies lexicalized with lemmas (*"piano - dobj - play", "man - nsubj - play"*), their unlabeled versions (*"piano - play", "man - play"*) and the versions with super-sense lexicalization (*"noun.artifact - dobj - verb.competition", "noun.person - nsubj - verb.competition"*) and (*"noun.artifact - verb.competition", "noun.person - verb.competition"*). The same is done with the semantic role labeling annotations, the possible chains are with argument labels with lemma lexicalization (*"play - A0 - man", "play - A1 - piano"*), without argument labels with lemma lexicalization (*"play - man", "play - piano"*), with argument labels and super-sense lexicalization (*"verb.competition - A0 - noun.person", "verb.competition - A1 - noun.artifact"*) and without argument labels with super-sense lexicalization (*"verb.competition - noun.person", "verb.competition - noun.artifact"*).

In order to compare and assess how linguistically similar a question is to candidate answer, we obtain the chains at different lexicalization level for both them and then apply a similarity metrics to the obtained chains.

For example, we want to compare the question *"Is Guinness a kind of beer?"* with the passage *"Guinness produces different kinds of beers"*. We extract the chains of lemma bigrams (excluding stopwords) for the question and we obtain *[be_guinness, guinness_kind, kind_beer]*. We do the same for the passage and we obtain *[guinness_produce, produce_different, different_kind, kind_beer]*. A simple similarity metric could be the number of common tokens, in this case we have one common tokens *kind_beer*.

Next, we list all the similarity metrics that we apply to the chains.

Overlap The overlap features count the ratio of tokens in common between the question and the answer as $\frac{|t_q \cap t_a|}{|t_q|}$, where t_q is the set of tokens belonging to the question and t_a the set of tokens belonging to the answer.

With this simple overlap formula we calculate the overlap of unigrams with all the different lexical levels, resulting in 6 features. Other 15 features are obtained calculating the overlap of 2-grams, 3-grams and 4-grams of all the lexicalizations except for the named entities, as they are already n-grams of words in most of the cases.

We also calculate the overlap of the dependency chains and semantic role labeling chains, both labeled and unlabeled and both with lemma and super-sense lexicalizations, resulting in 8 features.

For the different lexicalizations of the unigrams we also calculate the Jaccard Index as $\frac{|t_q \cap t_a|}{|t_q \cup t_a|}$ resulting in additional 6 features. We do not calculate the Jaccard index for the n-grams and for the dependency and semantic role labeling chains because of their sparsity.

The 35 overlap features are shown in Table 6.4.

Group ls	**Subgroup** overlap
Overlap of lemmas	
Overlap of concatenations of lemmas and PoS tags	
Overlap of named entities	
Overlap of stems	
Overlap of super-senses	
Overlap of terms	
Overlap of labeled dependencies with lemma lexicalization	
Overlap of labeled dependencies with super-sense lexicalization	
Overlap of unlabeled dependencies with lemma lexicalization	
Overlap of unlabeled dependencies with super-sense lexicalization	
Overlap of labeled semantic roles with lemma lexicalization	
Overlap of labeled semantic roles with super-sense lexicalization	
Overlap of unlabeled semantic roles with lemma lexicalization	
Overlap of unlabeled semantic roles with super-sense lexicalization	
Jaccard Index of lemmas	

Jaccard Index of concatenations of lemmas and PoS tags
Jaccard Index of named entities
Jaccard Index of stems
Jaccard Index of super-senses
Jaccard Index of terms
Overlap of lemma bigram s
Overlap of bigrams of concatenations of lemmas and PoS tags
Overlap of stem bigrams
Overlap of super-sense bigams
Overlap of term bigrams
Overlap of lemma trigram s
Overlap of trigrams of concatenations of lemmas and PoS tags
Overlap of stem trigrams
Overlap of super-sense trigams
Overlap of term trigrams
Overlap of lemma tetragram
Overlap of tetragrams of concatenations of lemmas and PoS tags
Overlap of stem tetragrams
Overlap of super-sense tetragams
Overlap of term tetragrams

Table 6.4: Overlap features

Frequency We use standard Information Retrieval techniques to obtain a measure of similarity between question and answer that takes into account the frequency of the tokens in the texts and in the whole corpus. We assign scores to the question-answer pairs according to the Tf-Idf weighting scheme, to the BM25 weighting scheme and to the Language Modeling (with Dirichlet priors [Zhai and Lafferty, 2001]) for all the different lexicalization levels except for the named entities, for a total of 15 features. The features are shown in Table 6.5.

Group ls	**Subgroup** frequency
BM25 with lemmas
BM25 with concatenations of lemmas and PoS tags
BM25 with stems
BM25 with super-senses
BM25 with terms
Language Modeling with lemmas
Language Modeling with concatenations of lemmas and PoS tags
Language Modeling with stems
Language Modeling with super-senses
Language Modeling with terms

TF-IDF with lemmas
TF-IDF with concatenations of lemmas and PoS tags
TF-IDF with stems
TF-IDF with super-senses
TF-IDF with terms

Table 6.5: Frequency features

Density We adopted a slight modification of the Minimal Span Weighting proposed by Monz [2004], calculating it for all the different lexicalization levels. This gave us 6 features.

The original formula contains three components: text similarity, span size ratio and matching term ratio. The text similarity intercepts global similarity, the span intercepts local similarity and the matching term ratio counterbalances the local similarity, i.e. in the case of only one term matching out 5 question terms the span part of the formula would return a value of 1, while the matching term would be $\frac{1}{5}$. For such a reason, in order to obtain a high local similarity, the highest number of terms from the question should be present in the smallest span of terms in the answer.

As we have other features, like the frequency ones, that address for the text similarity, we retained only the local similarity part, resulting in the following formula:

$$\left(\frac{\mid t_q \cap t_a \mid}{1 + \max(mms) - \min(mms)}\right)\left(\frac{\mid t_q \cap t_a \mid}{\mid q \mid}\right)$$

where t_q and t_a are the sets of tokens respectively of the question and the answer; $\max(mms)$ and $\min(mms)$ are the initial and final location of the shortest sequence of answer tokens containing all the question tokens.

The features are shown in Table 6.6.

Group ls	**Subgroup** density
Density of lemmas	
Density of concatenations of lemmas and PoS tags	
Density of named entities	
Density of stems	
Density of super-senses	
Density of terms	

Table 6.6: Density features

Machine Translation Research in machine translation, a sub-field of computational linguistics, investigates th use of computational methods in order to

translate text from one language to another. Due to the availability of aligned corpora, statistical approaches to MT have rapidly grown in the last decade, leading to better phrase-based translations.

The objective of machine translation in CQA is to "bridge the lexical chasm" between the question and the answer. We calculate the probability of the question being a translation of the answer $P(Q \mid A)$ and use it as a feature:

$$P(Q \mid A) = \prod_{q \in Q} P(q \mid A)$$

$$P(q \mid A) = (1 - \lambda) P_{ml}(q \mid A) + \lambda P_{ml}(q \mid C)$$

$$P_{ml}(q \mid A) = \sum_{a \in A} (T(q \mid a) P_{ml}(q \mid A))$$

where the probability that the question term q is generated from answer A, $P(q \mid A)$, is smoothed using the prior probability that the term q is generated from the entire collection of answers C, $P_{ml}(q \mid C)$ and λ is the smoothing parameter. $P_{ml}(q \mid C)$ is computed using the maximum likelihood estimator.

As the translation of a word to itself $P(w \mid w)$ is not guaranteed to be high, we set $P(w \mid w) = 0.5$ and re-scale $P(w'|w)$ for all the other w' terms in the vocabulary to sum up to 0.5, so that $\sum_{w' \in W}(w' \mid w) = 1$. This is needed for the adoption of translation models for retrieval tasks, as the exact world overlap of question and answer is a good predictor [Surdeanu et al., 2011].

Calculating the translation models for all the lexicalization degrees and for all the combinations of dependencies and semantic role labeling chains, we obtain 14 features. The features are shown in Table 6.7.

Group ls	**Subgroup** machine translation
Machine Translation of lemmas	
Machine Translation of concatenations of lemmas and PoS tags	
Machine Translation of named entities	
Machine Translation of stems	
Machine Translation of super-senses	
Machine Translation of terms	
Machine Translation of labeled dependencies with lemma lexicalization	
Machine Translation of labeled dependencies with super-sense lexicalization	
Machine Translation of unlabeled dependencies with lemma lexicalization	
Machine Translation of unlabeled dependencies with super-sense lexicalization	
Machine Translation of labeled semantic roles with lemma lexicalization	
Machine Translation of labeled semantic roles with super-sense lexicalization	
Machine Translation of unlabeled semantic roles with lemma lexicalization	
Machine Translation of unlabeled semantic roles with super-sense lexicalization	

Table 6.7: Machine Translation features

Others We consider 4 additional miscellaneous features: the length of the exact overlap of the sequences of words in the question and the answer normalized by the length of the question, the length ratio of the question and the answer, the inverse of the length of the answer and the inverse of the length of the question. The features are shown in Table 6.8.

Group ls	**Subgroup** other
Number of consecutive overlapping words	
Length of the answer over the length of question (in characters)	
1 over the length of the answer	
1 over the length of the question	

Table 6.8: Other features

Distributional Semantics

We already introduced the DSMs, their construction and the way they are used as features in Chapter 3.

Group ls	**Subgroup** distributional semantics
Semantic similarity using the LSA on Wikipedia corpus	
Semantic similarity using the Random Indexing on Wikipedia corpus	
Semantic similarity using the LSA after Random Indexing on Wikipedia corpus	
Semantic similarity using the Continuous Skip-gram Model on Wikipedia corpus	
Semantic similarity using the LSA on Yahoo! Answers corpus	
Semantic similarity using the Random Indexing on Yahoo! Answers corpus	
Semantic similarity using the LSA after Random Indexing on Yahoo! Answers corpus	
Semantic similarity using the Continuous Skip-gram Model on Yahoo! Answers corpus	

Table 6.9: Distributional-Semantics-based features

For computing the Distributional Semantics (ds) features for this set of experiments, we construct the $\mathbf{M}$ matrix both using Wikipedia as a corpus and using the set of all the answers in the training set obtained from the *Yahoo! Answers 2011 dataset* that we use for the evaluation (see Section 6.4 and Section 6.6). We do so to use both general purpose texts incorporating common

sense knowledge and knowledge that is specific to the dataset we want to actually use.

We exploit three methods for approximating the original matrix by rank reduction: Latent Semantic Analysis (LSA) [Deerwester et al., 1990], Random Indexing (RI) [Kanerva, 1988] and LSA over RI ($LSARI$) [Sellberg and Jönsson, 2008]. We furthermore adopted the Continuous Skip-gram Model (CSG) [Mikolov et al., 2013b] for learning word vector representations. The aim of these methods varies from discovering high-order relations between entries to improving efficiency by reducing its noise and dimensionality. The number of dimensions of the vector representations for all the methods is 400, stopwords are removed and only unigrams are considered.

We represent the question and the answers as the sum of the vectors of the terms they contain and then we calculate the cosine similarity of the two resulting vectors. We calculate the cosine similarity scores using vectors from the three types of semantic spaces constructed on both the corpora, resulting in 8 features.

The features are shown in Table 6.9.

6.3.2 User Features

A considerable part of the features are related to the user-centric activity, to capture their behavior and history. The question and answer history and some standard fields from the public profile description are usually available in all major CQA platforms. We also assume that questions are tagged with a category, which is the case for most of the communities that enforce a strict category systems or allow the possibility of collaborative tagging. Although most of the features we present here have been used in prior literature of best answer selection [Agichtein et al., 2008], the decomposition of the same features across different question categories has never been explored in this context. The subgroups of user features are summarized next.

User Profile The user profile contain information that might be a good proxy for the level of user's involvement in the community. These include: the presence of a resume, of a textual self-description of the user, of a title and a profile picture (surprisingly, a remarkably good estimator of expertise [Gînsca and Popescu, 2013]) and the amount of time the user has been registered on the platform at the time the question was asked (we refer to it as *age* for simplicity), for a total of 5 features. The features are shown in Table 6.10.

Group u	**Subgroup** profile
Presence of a resume in the user profile (1 if present, 0 otherwise)	
Length of the resume (in characters)	

Presence of a title in the user profile (1 if present, 0 otherwise)
Presence of a picture in the user profile (1 if present, 0 otherwise)
Time since the account creation

Table 6.10: User Profile features

Question and Answers The number of questions the user asked, deleted, answered, flagged, starred, and their normalized versions by user age are the basic gages for user activity. In addition to that, we also compute the ratio of those values divided by all the questions asked. We replicate the same features we calculated on the questions asked by the user also on the answers given by the user, adding also features about the thumbs up and down received by the answers and their ratio and delta. Overall, we define 19 features for the questions and 19 for the answers. The features are shown in Table 6.11.

Group u	**Subgroup** question answer
Number of (not deleted) questions asked by the user	
Number of deleted questions asked by the user	
Number of answered questions asked by the user	
Number of flagged questions asked by the user	
Number of questions with a star asked by the user	
Number of (not deleted) questions asked by the user over the time since the account creation	
Number of deleted questions asked by the user over the time since the account creation	
Number of answered questions asked by the user over the time since the account creation	
Number of flagged questions asked by the user over the time since the account creation	
Number of questions with a star asked by the user over the time since the account creation	
Number of (not deleted) questions over all the questions asked by the user	
Number of deleted questions over all the questions of the user	
Number of answered questions over all the questions asked by the user	
Number of flagged questions over all the questions asked by the user	
Number of questions with a star over all the questions asked by the user	
Minimum Automatic Readability Index of questions asked by the user	
Maximum Automatic Readability Index of questions asked by the user	
Average Automatic Readability Index of questions asked by the user	
Number of questions over number of answers given by the user	
Number of (non deleted) answers given by the user	
Number of deleted answers given by the user	

Number of best answers given by the user
Number of flagged questions asked by the user
Number of (not deleted) answers given by the user over the time since the account creation
Number of deleted answers given by the user over the time since the account creation
Number of best answers given by the user over the time since the account creation
Number of flagged answers given by the user over the time since the account creation
Number of (not deleted) answers over all the answers given by the user
Number of deleted answers over all the answers given by the user
Number of best answers over all the answers given by the user
Number of flagged answers over all the answers given by the user
Number of positive votes that the answers given by the user have received
Number of negative votes that the answers given by the user have received
Difference of positive and negative votes that the answers given by the user have received
Number of positive votes over number of negative votes that the answers given by the user have received
Minimum Automatic Readability Index of answers given by the user
Maximum Automatic Readability Index of answers given by the user
Average Automatic Readability Index of answers given by the user

Table 6.11: Question Answer features

Question Categories We replicate the same features defined for the question and answer history of the user, but considering only the category of the question actually asked. For example, if the question belongs to the category "sports" we count the questions asked and the answers given by the users in that category. This will help us estimate the user expertise and how much the user is engaged in the specific topic rather than his generic expertise or interest in different topics than the one the asker is interested in.

So we add additional 19 features for questions in the category and other 19 for the answers in the category.

We also add 3 additional features that consider the entropy H of discrete probability distribution p obtained by counting the number of questions, the number of answers and the combined number of question and answers in all the different categories ($\|p\|$ is the number of categories).

$$H(p) = -\sum_{i=0}^{\|p\|} p_i \log_2 p_i$$

This allows us to evaluate how specific (high entropy) or spread out (low entropy) the user knowledge (or interest) is. The features are shown in Table 6.12.

Group u	**Subgroup** category
Number of (not deleted) questions asked by the user in the category of the question	
Number of deleted questions asked by the user in the category of the question	
Number of answered questions asked by the user in the category of the question	
Number of flagged questions asked by the user in the category of the question	
Number of questions with a star asked by the user in the category of the question	
Number of (not deleted) questions asked by the user over the time since the account creation in the category of the question	
Number of deleted questions asked by the user over the time since the account creation in the category of the question	
Number of answered questions asked by the user over the time since the account creation in the category of the question	
Number of flagged questions asked by the user over the time since the account creation in the category of the question	
Number of questions with a star asked by the user over the time since the account creation in the category of the question	
Number of (not deleted) questions over all the questions asked by the user in the category of the question	
Number of deleted questions over all the questions of the user in the category of the question	
Number of answered questions over all the questions asked by the user in the category of the question	
Number of flagged questions over all the questions asked by the user in the category of the question	
Number of questions with a star over all the questions asked by the user in the category of the question	
Minimum Automatic Readability Index of questions asked by the user in the category of the question	
Maximum Automatic Readability Index of questions asked by the user in the category of the question	
Average Automatic Readability Index of questions asked by the user in the category of the question	
Number of questions over number of answers given by the user in the category of the question	
Number of (non deleted) answers given by the user in the category of the question	

Number of deleted answers given by the user in the category of the question
Number of best answers given by the user in the category of the question
Number of flagged questions asked by the user in the category of the question
Number of (not deleted) answers given by the user over the time since the account creation in the category of the question
Number of deleted answers given by the user over the time since the account creation in the category of the question
Number of best answers given by the user over the time since the account creation in the category of the question
Number of flagged answers given by the user over the time since the account creation in the category of the question
Number of (not deleted) answers over all the answers given by the user in the category of the question
Number of deleted answers over all the answers given by the user in the category of the question
Number of best answers over all the answers given by the user in the category of the question
Number of flagged answers over all the answers given by the user in the category of the question
Number of positive votes that the answers given by the user have received in the category of the question
Number of negative votes that the answers given by the user have received in the category of the question
Difference of positive and negative votes that the answers given by the user have received in the category of the question
Number of positive votes over number of negative votes that the answers given by the user have received in the category of the question
Minimum Automatic Readability Index of answers given by the user in the category of the question
Maximum Automatic Readability Index of answers given by the user in the category of the question
Average Automatic Readability Index of answers given by the user in the category of the question
Entropy of the vector constructed by counting the number of questions in each category
Entropy of the vector constructed by counting the number of answers in each category
Entropy of the vector constructed by counting the number of questions and answers in each category

Table 6.12: Category features

Behavioral Other features are related to the user behavior on the system. We count how many positive and negative votes are provided, plus their deltas and ratios, we measure the answering speed as the temporal gap between the time of the question and answer publications, and so on, for a total of 8 features. The features are shown in Table 6.13.

Group u	**Subgroup** behavioral
Internal Yahoo! Answer authority score of the user	
Number of flags given by the user	
Number of positive votes given by the user	
Number of negative votes given by the user	
Difference between the number of positive votes and the number of negative votes given by the user	
Number of positive votes over the number of negative votes given by the user	
Time between the question is posted and the answer is given by the user	
Number of answers given to this question	

Table 6.13: Behavioral features

6.3.3 Network Features

The network (n) features we propose arise from expert finding literature, where a content-agnostic analysis of the interactions between participants in CQA is shown help rank people by their general expertise in answering questions. For instance, users who provided high-quality answers (i.e., marked as best answers) to many questions, will likely provide good answers in future interactions as well. Also, the estimation of the users' expertise may not depend just on their direct interactions, but also from the interactions of other users, in a recursive fashion. For example one might imagine that, given a specific domain of knowledge, answering correctly a question made by an expert is a better indication of expertise than answering a question made by a newbie.

These considerations has motivated past research in the study of *Expertise Networks* [Zhang et al., 2007a], especially for CQA. Expertise Networks are weighted graphs where nodes are users and weighted edges model interactions that account for the flow of activity, knowledge or status differences among peers. In the past, three main Expertise Networks have been defined and studied for CQA. We provide visual examples for each in Figure 6.1.

The first is the *Asker Replier Network (ARN)* [Jurczyk and Agichtein, 2007], where directed edges flow from askers to answerers and are weighted by the number of replies. The second is the *Asker Best-Answerer Network (ABAN)* [Bouguessa et al., 2008, Gyongyi et al., 2007], where directed edges flow from askers to the best answerers and are weighted by the number of best answers

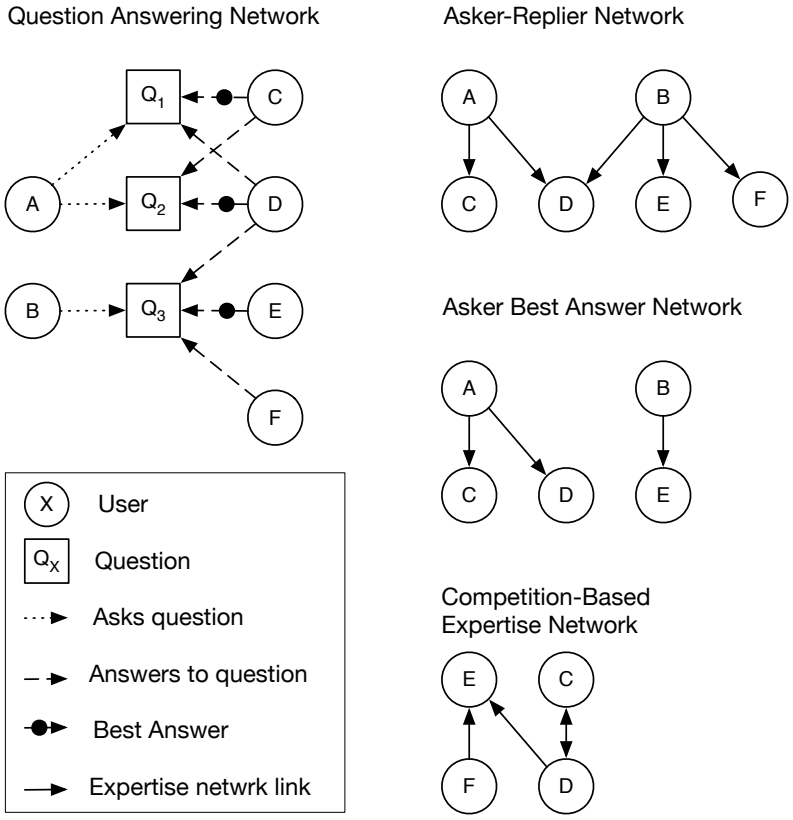


Figure 6.1: The graph of relations between askers, questions, and answerers (left) and the three types of Expertise Networks derived by it (right).

given. The last is the *Competition-Based Expertise Network (CBEN)* [Aslay et al., 2013], where edges flow between all the users who answered the same question towards the user who gave the best answer to that question; the possibility of building such a network is conditioned by the possibility for the users to explicitly mark the best answer, which is most often true in large scale CQAs. The advantage of ARN is that it needs less information to be built but, ignoring the signal coming from the best answer, it considers all the answers to have equal value. ABAN addresses this problem but on the other hand it disregards the information of people who answered and whose answer was not selected as the best. CBEN was proposed to take into account both aspects and to capture at the same time the inherent competition that exists between answerers to get awarded with the best answer. Also, no relation between asker and answerer is represented in CBEN under the assumption that asking a question is not necessarily related to a lack of expertise [Zhang et al., 2007a,b], especially in broad general purpose question answering communities.

The application of graph *centrality* metrics to the Expertise Networks mentioned above produces a ranking of the users based on their expertise. Depending on the specific combination of network and centrality, the ranking might convey different meanings, but in all the cases users with higher scores are supposed to have higher expertise compared to their peers with lower scores. In previous work, this assumption was validated by multiple experiments and some specific network-centrality combinations (PageRank on ARN, indegree on ABAN, HITS on CBEN) have proved to work best in the task of best answer prediction [Aslay et al., 2013]. In this work we aim to include in a learning to rank framework a wide set of features, therefore we do not restrict ourselves to specific pairs but we consider instead all the combinations of Expertise Networks (ARN, ABAN, CBEN) with the centrality metrics that have been applied to them in past work (PageRank [Page et al., 1999], HITS [Kleinberg, 1999], indegree) for a total of 9 features. We consider networks built on the full question-answer dataset with no distinction of topic, as we want to measure general expertise with network features and account for relevance with the textual features. Such features are shown in Table 6.14.

Group n	**Subgroup** arn - aban - cben
Indegree of the user in the Asker Replier Network	
PageRank of the user in the Asker Replier Network	
Hits Authority of the user in the Asker Replier Network	
Indegree of the user in the Best-Answerer Network	
PageRank of the user in the Best-Answerer Network	
Hits Authority of the user in the Best-Answerer Network	
Indegree of the user in the Competition-Based Expertise Network	
PageRank of the user in the Competition-Based Expertise Network	
Hits Authority of the user in the Competition-Based Expertise Network	

Table 6.14: Network features

6.4 Dataset

The instance of CQA we consider for our experiments is Yahoo! Answers, because of its popularity and richness of content. Launched in 2005, it is one of the largest general purpose CQA services to date, hosting questions and answers on a broad range of topics, categorized through a predefined two-level taxonomy. There are 26 predefined Top-Level Categories (TLC), such as Politics, Sports or Entertainment, and a growing number of Leaf-Level Categories (LLC) –more than 1,300 at the time of this study– such as Makeup or Personal Finance. Similarly to other CQA portals, Yahoo! Answers follows a strict question-answer

format, with questions submitted as short statements with optional detailed description, and a mandatory leaf-level category that is assigned by the asker. Questions have a lifecycle of states that goes from *open*, to *voting* and finally to *resolved*, and users can actively moderate content using several feedback mechanisms, such by marking spam or abusive content, adding *stars* to interesting questions, voting for best answers, and giving *thumbs-up* or *thumbs-down* ratings to answers. Among all the feedback signals, the most important is the selection of the *best answer*, which is designated by the asker or, if the asker does not provide it after a given time, it is selected by the community with majority vote. The process of best answer selection is important not only to reward contributors according to the Yahoo! Answers incentive scheme[1], but also for archival purposes, as the best answer will be given evidence in the page and will serve users who might have the same question in the future.

6.4.1 Yahoo! Answers 2011

We first collected a data sample from Yahoo! Answers related to the period between January and December 2011, for a total of $>$ 7.2M resolved questions with best answer assigned by the asker, $>$ 39.5M answers and $>$ 6.1M unique users. The dataset contains the text of the question and answers, their metadata (timestamp, question category, number of thumbs up and down, best answer mark) and the metadata associated to the user involved in the process (user self-description, subscription date, number of questions asked and answers given, number of best answers, presence of thumbnail photo in the profile). Each question has only one answer marked as the best one.

As Yahoo! Answers is a general purpose portal, not only it covers different topics but it also hosts a broad variety of question *types*.

In practice, every forum category has some mix of requests for factual information, advice seeking and social conversation or discussion [Harper et al., 2009]. The most refined categorization obtained on Yahoo! Answers so far has been proposed by Aslay et al. [2013], who extended the seminal work by Adamic et al. [2008] and used k-means to cluster Yahoo! Answers leaf level categories using features such as the average number of replies to a question and the average number of characters in a reply, and some activity-based features such as the proportion of questions with contradictory answer ratings (thumbs up vs. thumbs down). The optimal R^2 was obtained for $k = 4$, corresponding to the following main question types: *factual-information seeking* (31% of the questions), *subjective-information seeking* (32%), *social discussion* (10%), and

[1] A new user is granted 100 points and asking a question costs 5 points. Several user actions are worth new points, among which the submission of an answer that is the most rewarding one (, as it is worth ()10 points). Detailed scheme available at: `http://answers.yahoo.com/info/scoring_system`

poll-survey conducting (27%). We use this categorization to compare the feature performance also across question types.

6.4.2 Yahoo! Answers Manner Questions

To compare our results directly against some state-of-the-art methods, we decided to replicate the experiments with a freely available dataset [2] that contain a sample of manner questions collected from the US Yahoo Answers site. Manner questions are those questions that ask how to do something. Following what was done in previous work [Surdeanu et al., 2011], the manner questions are extracted following two simple heuristics that aim at preserving only high quality questions and answers. This is done by retaining all the questions that i) match the regular expression: `how (to | do | did | does | can | would | could | should)`, and ii) have at least four words, out of which at least one is a noun and at least one is a verb. This process yields 142,627 questions and 771,938 answers, with an average of 5.41 answers for each question.

6.5 Experimental Setup

Next we describe the problem under study and the framework we use to address it, along with four baselines we compare our method against.

Problem Statement

Given in input a question q and the set of its answers $A(q)$, among which exactly one answer $a^* \in A(q)$ has been selected as best answer, output a rank of the answers in the set $A(q)$ that has a high likelihood of a^* being placed high in the rank. This problem is a generalization of the best answer selection, and can be reduced to it if only the first element in the ranking is considered, but allows a more detailed analysis of the results and a richer comparison between methods.

6.5.1 Learning to Rank for Best Answer Prediction

We already introduced the Learning to Rank setting in Chapter 4. We opted for Random Forests (RF) [Breiman, 2001] because of its resilience to overfitting, a problem that may affect our experimental setting due to the size of our dataset, and because of the successful results in several use cases related to CQA [Dalip et al., 2013] and in other large scale retrieval experiments [Mohan et al., 2011].

[2] `http://webscope.sandbox.yahoo.com/catalog.php?datatype=l`

System Configuration

With respect to the general architecture we described in Chapter 2, the configuration of the system consists of:

- all the English NLP analyzers (except Word Sense Disambiguation, as it was too computationally expensive to carry out on the large-scale dataset we are adopting),
- all the different search engines (their score are used for calculating the frequency features described in Section 6.3.1,
- all features extractors presented in Section 6.3,
- the Random Forests implementation described in Section 4.4 as ranker.

Baselines

We compare our approach with four different baselines.

- *BM25.* Standard ranking function used in information retrieval to rank matching documents according to their relevance to a given search query. We consider the question as query and the answers as documents. We chose this baseline over other IR baselines because it is the best performing one in our dataset.
- *Finding high-quality content in social media* Agichtein et al. [2008]. A supervised method trained on measures of Text Quality such as grammatical, syntactic and semantic complexity, punctuation and typo errors, along with simple question-answer similarity and user expertise estimations. Readability and informativeness are also included. Their best performance was achieved using Stochastic Gradient Boosted Trees. We replicated their learning approach and feature set, as the dataset they adopted for the experiments was made of 8,366 question/answer pairs so their results were not directly comparable with ours. We chose this baseline because it was the state of the art on Yahoo! Answers data.
- *Learning to Rank Answers* Surdeanu et al. [2011]. Combines linguistic features, those based on translation, classical frequency, density ones and web-correlation based ones with a learning to rank approach, carried out with an averaged perceptron. It was applied on the *Yahoo! Answers Manner Questions dataset* as a testbed. The authors did not use any user-based feature nor expertise-based ones as this kind of information is missing from the dataset, but they also did not adopt Text Quality features that we adopt and the levels of lexicalizations of their linguistic

features are only terms, lemmas and super-senses. We chose this baseline because it was the state of the art on *Yahoo! Answers Manner Questions dataset* for P@1.

- *Improved answer ranking* Hieber and Riezler [2011]. Similar to the previous one, this work relies mainly on textual features, but adopting Piggybacking features on web snippets. The ranking is done adopting a SVM-based ranker. Their evaluation was carried out on *Yahoo! Answers Manner Questions dataset* as well. We chose this baseline because it was the state of the art on *Yahoo! Answers Manner Questions dataset* for MRR.

6.6 Experimental Results

We evaluate our learning to rank framework splitting the dataset into 70% training, 10% development and 20% test, using a temporal criterion (older questions for training). The 10% development was needed for tuning machine translation parameter λ (see Section 6.3.1). For each question, all its answers are ranked by the learning to rank method. To allow a direct comparison of the quality of the ranking with results in previous work, we use three standard IR metrics that have been commonly used to evaluate this task, namely *Mean Reciprocal Rank (MRR)*, *Precision at 1 (P@1)* and *Discounted Cumulative Gain (DCG)*. When considering the answers to a single question, these are formally defined as follows:

$$\text{RR} = \frac{1}{\text{rank(BA)}} \quad \text{DCG}_\text{k} = \sum_{i=1}^{k} \frac{2^{rel_i}-1}{\log_2(i+1)} \quad \text{P@1} = rel_1$$

where A is the set of answers, $rank(BA)$ is the rank of the best answer for that question, and rel_i is an indicator function of relevance that returns 1 if the answer in the i^{th} position in the ranking is the best answer. All the scores are then averaged over all the questions ($\frac{1}{|Q|}\sum_{q \in Q} score(q)$). In case the best answer is ranked first, $MRR = DCG = P@1 = 1$. As each question has only one answer marked as correct (the best answer) the $DCG = nDCG$, because the ideal DCG is equal to 1. All differences have to be considered statistically significant (using the non-parametric Randomization test, as suggested in [Smucker et al., 2007], with $p < 0.01$) unless otherwise specified.

6.6.1 Performance Analysis (Yahoo! Answers 2011)

To gain insights about the predictive power of different feature families, we train the model on several subsets of features, with a *greedy* selection procedure. We first separately test each family and pick the best performing one; at the next step, we keep that family and combine it with all the others to select the best

Features	P@1	MRR	DCG
BM25	0.4143	0.5532	0.6567
Agichtein et al. [2008]	0.5243	0.6375	0.6962
tq	0.5305	0.7016	0.7655
ls	0.5143	0.6921	0.7613
ds	0.4782	0.6760	0.7564
u	0.5218	0.7009	0.7757
n	0.4527	0.6645	0.7484
tq+u	0.6201	0.7597	0.8260
tq+n	0.5862	0.7366	0.8080
tq+ds	0.5536	0.7144	0.7910
tq+ls	0.5515	0.7129	0.7897
tq+u+n	0.6416	0.7742	0.8370
tq+u+ds	0.6210	0.7606	0.8266
tq+u+ls	0.6199	0.7597	0.8260
tq+lo+ds	0.5519	0.7143	0.7901
tq+u+n+ds	0.6450	0.7752	0.8379
tq+u+n+ls	0.6414	0.7739	0.8368
all	0.6471	0.7798	0.8389

Table 6.15: Predictive power of the learning to rank framework trained on different feature subsets, on the *Yahoo! Answers 2011 dataset*.

Feature families are Text Quality (tq), Linguistic Similarity (ls), Distributional Semantics (ds), User (u), and expertise network (n). Best feature combinations in each section of the table are highlighted in bold.

combination. The process is repeated until all the feature families are included. The greedy strategy allows us to find a locally optimal choice at each stage, with the hope of finding a global optimum in a reasonable time. Results are shown in Table 6.15.

The most predictive features are the ones belonging to the **tq** family. This group includes 44 features that capture many facets of the text structure that are indeed good proxies for the answer quality. On the other hand, **n** features alone are the worst performing; this is expected as centrality metrics capture general expertise in a content-agnostic way, so they do not embed information about the topic or structure of the questions and answers. A similar consideration can be done for the user features even though their performance is sensibly higher than the network features. This supports the findings in previous work [Chen and Nayak, 2008], that found simple user features such as the percentage of best answers very predictive of the level of user expertise. Finally, **ls** features

Feature	Δ
tq: Preposition Count	0.049
tq: Verbs not in Question	0.045
tq: Nouns not in Question	0.045
tq: Unique Words in Answer	0.043
tq: Pronouns Count	0.042
tq: Punctuation Count	0.039
tq: Average Words per Sentence	0.039
ds: Random Indexing on Yahoo! Answers	0.039
ls: Super-senses Overlap	0.038
tq: Adjectives not in Question	0.036
tq: Conjunctions Count	0.035
tq: Capitalized Words Count	0.035
tq: "To be" Count	0.035
ls: Lemma Overlap	0.034
ls: Stem Overlap	0.034
ls: Term Overlap	0.032
tq: Auxiliary Verbs Count	0.034
ls: Super-senses BM25	0.031
n: Indegree on CBEN	0.030
u: Answerer's Best Answer Ratio	0.030

Table 6.16: Ablation test.

Δ measures the loss of performance in MRR when the feature is removed, when the full set of features is employed. Prefixes in names indicate the family of the feature.

outperform the **ds** features, when used in isolation; this may be mainly due to the very different dimensionality of the feature sets as Distributional Semantics include a set of just 6 features. Regarding the baselines, we note, as expected, that an approach that is not specifically tailored on the task like BM25 performs poorly. The method from Agichtein et al. [2008] has also a performance that is lower than the ones obtained by the single feature families partially because of the different training procedure but mainly because it is trained with a set of features that is smaller than the ones we consider inside each family.

When combining features in pairs, interesting patterns emerge. Even though **tq** and **ls** are the best performing individually, their combination improves the performance only slightly as the signal they bring is very overlapping. Indeed, their combination is the worst performing among all the feature pairings. The same happens with **ds** features. **n**, and especially **u** features, are instead more

orthogonal to the **tq** information and are able to boost the performance considerably. Most importantly, we find that **n** and **u** features carry predictive information that is non-overlapping, as the combination of both with **tq** features results in further noticeable improvement.

Combinations of three feature groups or more make clear that, despite the high informativeness on their own, the **ls** features give a fairly small contribution to the performance and replacing them with **ds** features leads even to a small improvement. Given that the time of computation of the **ls** features is roughly *12 times* more than the **ds** ones (as empirically measured in our test), it appears that **ds** features are stronger and more lightweight (they are very few) and therefore are more viable alternative.

The *MRR* score obtained with the combination of all the feature groups is a **22%** improvement over the baseline, while the *P@1* score is a **23%** improvement and *DCG* score is a **21%** improvement.

Besides the greedy aggregation of feature families, to discover which single features give the best signal for the prediction, we run an *ablation* test to measure the performance decrease Δ in the prediction when single features are removed from the set. The 20 ones with the highest values of Δ are reported in Table 6.16. We note that, although **tq** features tend to dominate, one feature from **ds** and one from **n** make it into the top 20 (8^{th} and 19^{th}, respectively).

As final remark, we note that when plotting the MRR and DCG for rankings that include the top n results only (Figure 6.2) we see that the values tend to increase considerably in the first positions of the ranking, meaning that the best answer, if not ranked as first, is usually ranked among the top 2 or 3 answers.

6.6.2 Detailed Analysis of Proposed Features

Feature	Rank
ds: Random Indexing on Yahoo! Answers	8
ds: Continuous Skip-gram Model on Yahoo! Answers	30
ds: LSA on Wikipedia	37
ds: LSA after Random Indexing on Wikipedia	38
ds: Continuous Skip-gram Model on Wikipedia	39
ds: Random Indexing on Wikipedia	40
ds: LSA after Random Indexing on Yahoo! Answers	89
ds: LSA on Yahoo! Answers	90

Table 6.17: Distributional-Semantics-based features ablation ranking

We analyzed in more the detail the results of the ablation test, focusing on the newly proposed features.

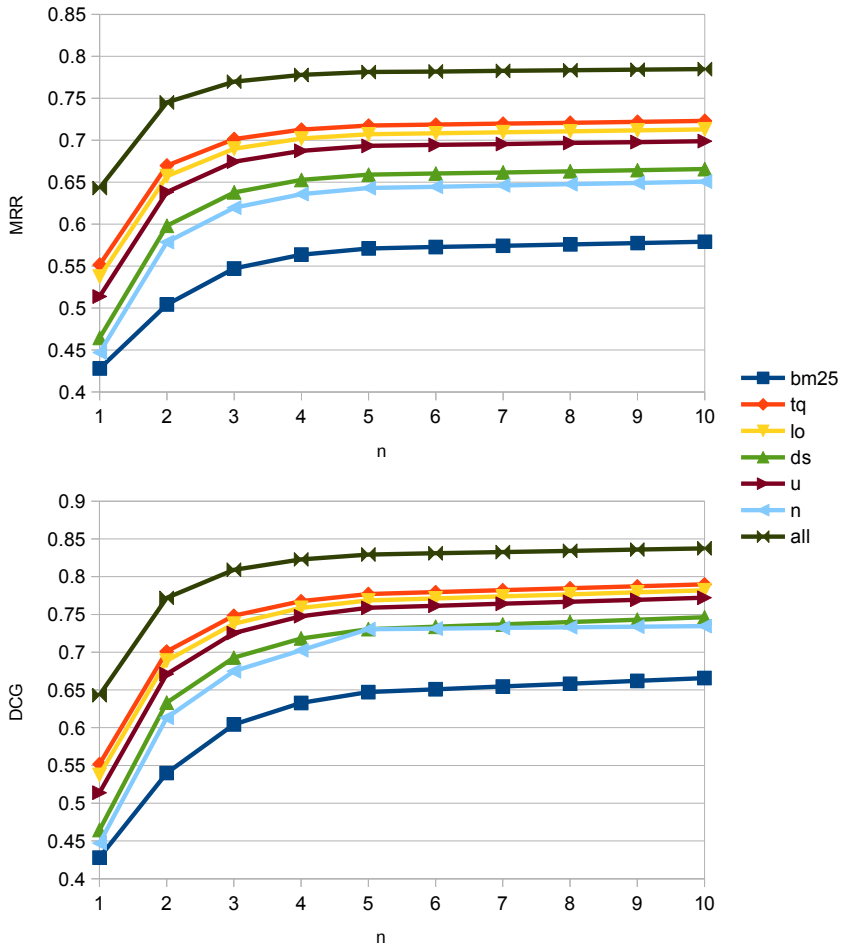


Figure 6.2: MRR and DCG computed for the first n positions of the ranking, for the different features families, plus the BM25 baseline and the full set of features.

Feature	Rank
n: Indegree on CBEN	19
n: Hits on CBEN	32
n: Indegree on ABAN	101
n: Hits on ABAN	108
n: Indegree on ARN	161
n: Hits on ARN	164
n: PageRank on ARN	170
n: PageRank on CBEN	183
n: PageRank on ABAN	184

Table 6.18: Network features ablation ranking

Considering the features based on Distributional Semantics (**ds**), reported in Table 6.17, we can clearly see that the best performing feature, Random Indexing on Yahoo! Answers, ranks 8-th. This is encouraging and suggests that the adoption of textual data coming from the dataset itself is helpful. Continuous Skip-gram Model on the same datasets is the second best one, ranking 30-th, supporting the suggestion of the Random Indexing feature. The other two features using models learned on the same dataset rank 89-th (LSA over Random Indexing) and 90-th (LSA), almost in the middle of the ranking. The difference with respect to Random Indexing suggests that probably the number of dimensions (400) is not an appropriate choice for the LSA, and an optimization of this parameter could lead to improvements.

The features that adopt Wikipedia as a text source for learning the models rank really close: 37-th for LSA, 38-th for LSA over Random Indexing, 39-th for Continuous Skip-gram Model and 40-th for Random Indexing. This suggests that the differences in models, in this case, are less influent than the dataset itself. As Wikipedia contains more than 4 million articles, the huge quantity of text in this dataset leads to similar behaving models.

Considering the Network based features (**n**), reported in Table 6.18, the best performing network structure is the Competition-Based Expertise Networks. Two features based on models calculated on this network are the top ranked: Indegree on CBEN is 19-th and Hits on CBEN is 32-nd. The same two models calculated on the Asker Best-Answerer Network are ranked in the middle of the ranking, 101-st and 108-th respectively, while those calculated on the Asker Replier Network are ranked lower in the ranking, 161-st and 164-th. The fact that both models, the simple indegree and the Hits authority, are found really close in the ranking suggests that they behave in a really similar way. At the bottom of the ranking we found the PageRank model calculated on ARN (170-th), on CBEN (183-rd) and ABAN (184-th). This suggests that PageRank is

not a good fit in this setting and leads to quite bad results.

6.6.3 Performance Analysis (Yahoo! Answers Manner Questions)

Features	P@1	MRR	DCG
BM25	0.4112	0.5606	0.6121
Surdeanu et al. [2011]	0.5091	0.6465	-
Hieber and Riezler [2011]	0.4844	0.6676	-
ds	0.6118	0.7689	0.8198
ls	0.618	0.7717	0.8236
tq	0.6245	0.7857	0.8352
ds+ls	0.618	0.7721	0.8236
ds+tq	**0.6532**	0.7920	0.8421
ls+tq	0.6401	0.7855	0.8352
ds+ls+tq	**0.6532**	**0.7922**	**0.8425**

Table 6.19: Predictive power of the learning to rank framework trained on different feature subsets, on the *Yahoo! Answers Manner Questions dataset*.

The last two baselines we consider (Surdeanu et al. [2011] and Hieber and Riezler [2011]) have been applied to the smaller *Yahoo! Answers Manner Questions dataset* described in Section 6.4. To get a fair comparison with them, we replicate their same experimental setup on the same dataset, and repeat the greedy feature family combination as described before. A Random Forest model is learned for each feature set, performances are reported in Table 6.19. Differently from the previous dataset, we do not perform a temporal split as timestamps are not available in the dataset, so we perform 5-fold cross validation with a 70-10-20 split (10% being a validation set).

All the three groups improve over the baseline significantly both in *P@1* and *MRR*, with **tq** being the most effective. It is worth noticing that the distributional semantics-based features alone (8 features) can compete with the other two groups of features, which are composed of 42 features for **tq** and 74 for **ls**.

Taking into account the combinations of features we observe that the best performing one is the composition of **ds** and **tq**. The combinations of **ds** and **ls** does not improve at all for *P@1* and improves just of 0.004 for *MRR* over the **ls** group alone, a non statistically significant improvement. This is expected as both groups try to intercept the topical similarity between question and answer.

The most interesting result that can be observed is that adding the **ls** group to the previous best scoring group **ds+tq** does not improve the performances

at all for *P@1* and improves just of 0.002 for **MRR** and 0.004 for **DCG**, again a non statistically significant improvement. This finding suggests that in this setting the linguistic features, that requires a really expensive preprocessing to be computed, can be substituted with a single features based on Distributional Semantics of words without any loss of accuracy.

Finally, the best *P@1* scores obtained with the **ds+tq** and **ds+tq+ls** feature groups are a **27%** improvement over the state of the art (best of the three baselines), while the best *MRR* scores obtained with the **ds+tq+ls** features group are an improvement of **18%** over the state of the art.

6.6.4 Question Categories

Different types of questions may imply different notions of "high-quality" answer. To investigate this aspect, we get back to the bigger *Yahoo! Answers 2011 dataset* and we break down the performance of the different feature families by the four question categories we defined in Section 6.4. For brevity, we report the values for MRR only (P@1 and DCG follow the same trends) and limit the analysis to feature families taken in isolation.

	Factual	**Subjective**	**Discussion**	**Poll**
tq	**0.7329**	**0.7242**	0.6676	0.6762
ls	0.7243	0.7117	0.6482	0.6350
ds	0.6879	0.6738	0.6377	0.6498
u	0.7221	0.7118	**0.6724**	**0.6878**
n	0.7003	0.6953	0.6132	0.6214
all	0.8059	0.7898	0.7508	0.7644

Table 6.20: MRR scores obtained with single feature families on the *Yahoo! Answers 2011 dataset*.

In agreement with previous work [Aslay et al., 2013], the best answer is more difficult to predict for discussion and poll-type questions, as they are naturally less suited to expert ranking. Best answers for factual and subjective questions are better surfaced by the **tq** features, while the **u** features are dominating discussions and polls.

Focusing on the novel features we introduce, we note their complementary behavior, being **ds** better than **n** in polls and discussion (and even better than **ls** for polls) but worse in factual and subjective questions. Also, it is worth noting that **ds** has the smaller variance in performance across categories.

6.6.5 Different Algorithms

Our decision to use a pointwise approach like RF as ranking algorithm is based on the intuition that pairwise and listwise approaches are not likely to be more effective because of the presence of only one correct answer for each question in the dataset. This means that we have a number of equally wrong answers that we cannot distinguish based on their relevance to the answer, so the full list of answers is not likely to bring more information than the single answers.

Moreover, due to the large size of the dataset, we believed that the resilience to overfitting that characterize RF would have been a great advantage.

In order to assess in our intuition was likely to be true, we run the evaluations on the same datasets with the same features, but using different algorithms.

We chose Logistic Regression (LR) as an alternative pointwise approach because it was successfully adopted in large-scale real-world QA scenarios [Ferrucci, 2011]. For pairwise approaches we chose RankSVM [Joachims, 2002] as the algorithm to test against, as SVMs were shown to be effective on the same *Yahoo! Answers Manner Questions dataset* [Surdeanu et al., 2011]. Finally, for a listwise approach, we chose to test against ListNet [Cao et al., 2007]. All the algorithms are used with their default parameters from the adopted libraries (RankLib[3] and SVMLight[4]), without a specific hyperparameter tuning.

	LR	**RankSVM**	**ListNet**	**RF**
Manner	0.6952	0.7683	0.7520	**0.7922**
Factual	0.7407	0.7774	0.7626	**0.8059**
Subjective	0.7183	0.7640	0.7411	**0.7898**
Discussion	0.6881	0.7256	0.7059	**0.7508**
Poll	0.7027	0.7286	0.7312	**0.7644**
All	0.7165	0.7491	0.7466	**0.7798**

Table 6.21: MRR scores obtained with different Learning to Rank algorithms on the *Yahoo! Answers 2011 dataset*.

The results in Table 6.21 show only the trends for MRR using all the features, but the same trends are also present by changing the adopted feature set combination and metric. Logistic Regression is the worst performing algorithm on all the sets of questions, while among RankSVM and ListNet the difference is really small with RankSVM obtaining slightly higher results on all question sets but *Poll*. None of the alternative algorithms can reach the performance levels reached by RF in any of the question sets, and this gives some empirical evidence that our choice was reasonable.

[3] `http://sourceforge.net/p/lemur/wiki/RankLib/`

[4] `http://svmlight.joachims.org`

6.7 Summary

We contribute to bring order to the vast literature on the task of best answer selection by gathering the largest set of features considered for this task so far, grouped in five families, combining them with a learning to rank approach, and testing them on large datasets from Yahoo! Answers. We propose a new suite of Distributional-Semantics-based features, in combination with the textual signal and the information from several Expertise Networks. Besides being able to outperform the prediction ability of state-of-the-art methods up to 27% in P@1, our experiments allow us also to draw important conclusions about the impact of different features employed that have never been spell out in previous literature due to a lack of extensive and systematic feature comparison. We summarize our findings as follows.

- *Textual features are by far the ones with higher predictive potential*, compared to user-centric features or to the expertise network centrality scores. This is mainly due to the fact that the content of the question and answers (their topic and structure) are a more important source of information to determine the question-answering match rather than the expertise of the answerers. Those features are to prefer when dealing with factual-type questions.

- Among the textual features, *Text Quality and Distributional Semantics are in general to prefer to Linguistic Similarity.* We indeed found that Linguistic Similarity's signal is mostly captured by other features already. This is an important finding as Linguistic Similarity features have been used in a number of previous approaches but are roughly 12 times more computationally expensive than Distributional Semantics ones. This answers RQ2.

- *The new Distributional-Semantics-based approach we propose achieves surprisingly good results* considering the very small cardinality of its feature set. This answers RQ1.

- User and network features determine a considerable improvement over the textual-based features and their contribution is not completely overlapping, meaning that *considering network interaction rather than the individual user activity adds real value to the prediction.*

Part III

Applications

7

An Artificial Player for "Who Wants to Be a Millionaire?"

"Real stupidity beats artificial intelligence every time."
– Terry Pratchett, *Hogfather*

In this chapter we describe one application of our semantically enhanced QA framework to "Who Wants to Be a Millionaire?", a quiz where language understanding and wide knowledge are needed in order to play the game. We will describe how we exploit QA for this task, how we actually make the system play the game with all its specific rules. Finally, we will compare the performances with a baseline based on Google and with human players.

7.1 Introduction

The work on intelligent computer games has a long history and has been one of the most successful and visible results of Artificial Intelligence research [Maybury et al., 2006]. Indeed, today artificial systems are able to compete and sometimes challenge human players in several complex games. Most of these games are *closed world* ones, meaning that they have a finite number of possible choices, which allows the researchers to solve them formally, even though they are hard to play due to the exponential dimensions of the search spaces. A more challenging type of games is represented by *open world* games, such as sport games or crosswords: they are less structured and, moreover, both the states of the game and the actions of the player cannot be easily enumerated, making the search through the space of possible solutions practically unfeasible.

One of the most recent results in this field was the success of *Watson*, the open-domain question answering system built by IBM Research, which in February 2011 beat the two highest ranked players of the quiz show "Jeopardy!" [Ferrucci et al., 2013, 2010].

We are particularly interested to games related to human language. They are classified in *word games*, in which word meanings are not important, and *language games*, in which word meanings play an important role [Littman, 2000]. Language games generally require a wide linguistic and common sense knowledge. "Who Wants to Be a Millionaire?" (WWBM) is a perfect example of a language game in which the player provides an answer to a question posed in natural language by selecting the correct answer out of four possible ones. Even though the number of possible answers is limited to four, being able to successfully play this game heavily depends on the player's knowledge, her understanding of the questions and her ability to balance the confidence in the answer against the risk taken in answering.

This chapter describes the architecture of a *Artificial Player* for the WWBM game, which leverages Question Answering (QA) techniques and both Wikipedia and DBpedia [Bizer et al., 2009] open knowledge sources in order to incorporate the knowledge useful for playing the game. A preliminary work that describes the architecture of the artificial player is presented in [Molino et al., 2013a]. It has been extended in the following directions: the use of DBpedia; the *decision making* strategy integrated to manage the "lifelines" characterizing the game; the possibility to retire from the game; the use of machine learning techniques to improve the process of scoring the candidate answers to a question. Extended related work about question answering, answer validation and language games are also provided, along with more extensive experiments on both the Italian and the English versions of the game.

Motivated by the challenge to develop an effective artificial player for the WWBM game, in this chapter we address two research questions, RQ3 and RQ4:

- *RQ3. To what extent can a QA system be designed in a language-independent way, by preserving its effectiveness?*
  We cope with this question by proposing a general architecture of a QA and Answer Scoring (AS) framework which exploits resources or algorithms specifically designed for a given language exclusively for basic NLP operations, such as part-of-speech tagging or stemming. The QA framework leverages Wikipedia and DBpedia open knowledge sources, while the AS module supplies several criteria to score candidate answers and to effectively combine scores through machine learning techniques. In order to assess the effectiveness of the framework for at least two different languages, we performed experiments on English and Italian.

- *RQ4. Is it possible to develop an artificial player for the WWBM game able to outperform human players?*
  We address this question by comparing the accuracy of the human players against that of an artificial player built using the QA and AS framework in RQ1. We evaluate the ability of the artificial player to play the WWBM game with all its rules, i.e. usage of "lifelines", answering in a condition of uncertainty, retiring from the game by taking the earned money.

The remaining of the chapter is organized as follows: Section 7.2 describes the rules of the game, while related work in the areas of language games, QA over linked data and answer validation are presented in Section 7.3. The architecture of the artificial player, the details of the QA and AS modules, and the decision making strategy adopted to play the game are provided in Sections 7.4-7.7. Section 7.8 reports the results of an extensive evaluation performed on Italian and English versions of the game, before drawing the final conclusions in Section 7.9.

7.2 Rules of the Game

WWBM is a language game, broadcast by many TV channels in several countries, in which a player must correctly answer a series of 15 multiple-choice questions of increasing difficulty. Questions are posed in natural language and the correct answer is selected among four possible choices.

Figure 7.1 shows an example of the question *Who directed Blade Runner?*, and the four possible answers **A)** Harrison Ford **B)** Ridley Scott **C)** Philip Dick **D)** James Cameron. There are no time limits to answer the questions. Moreover, contestants read the question in advance, and then at any time they can decide whether to attempt an answer or quit the game by keeping the earned money. Each question has a certain monetary value (level 1:€500; level 2:€1,000; level 3:€1,500; level 4:€2,000; level 5:€3,000; level 6:€5,000; level 7:€7,000; level 8:€10,000; level 9:€15,000; level 10:€20,000; level 11:€30,000; level 12:€70,000; level 13:€150,000; level 14:€300,000; level 15:€1,000,000).

If the answer is correct, the player earns a certain amount of money and continues to play by answering questions of increasing difficulty until either she reaches the last question or she retires from the game by taking the earned money. There are three *guarantee points* where the money is banked and cannot be lost even if the player gives an incorrect answer to one of the next questions: 3,000, 20,000 and 1,000,000 Euros, corresponding to the milestone questions 5, 10, 15, respectively. At any point, the contestant may use one or more of three "lifelines", which provide her with some form of assistance:

- *50:50*: this lifeline removes two wrong answers, leaving the player with a binary choice between the correct answer and the incorrect one;

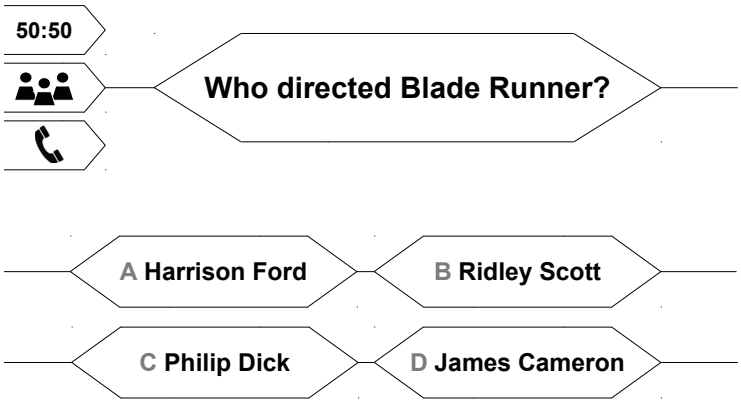


Figure 7.1: An example of "Who Wants to Be a Millionaire?" question.

- *Poll the Audience*: the player asks the studio audience to pronounce about the correct answer. The percentages of the audience for the 4 different answers are given to the player, who has the last word on the choice of the answer;
- *Phone a Friend*: the player has 60 seconds to phone a friend, and read the question with the four possible choices, in order to get a suggestion about the right choice.

The amount of earned money and the lifelines vary from country to country.

7.3 Background

Here we provide some related work in the areas of language games, QA over Linked Data and answer validation.

7.3.1 Language Games

Language games usually require a large amount of knowledge and deep reasoning capabilities to compete at human level. Artificial players for language games adopt Natural Language Processing (NLP) technologies in order to manage the complexity and ambiguity of the language, while storing and manipulating complex representations of the knowledge involved in the game.

A popular language game is solving crossword puzzles. Besides the linguistic knowledge, solving crosswords requires the satisfaction of constraints over the possible answers. The first experience reported in literature is *Proverb* [Littman et al., 2002], that exploits large libraries of clues and solutions to past crossword puzzles, while WebCrow [Ernandes et al., 2005], the first solver for Italian crosswords, exploits the Web as the main source of information, and a set of

previously solved games, as well. WebCrow is based on the sequential combination of "clue answering" and "grid filling", a solution which is radically different from a human approach. In order to find the best candidate words, WebCrow queries Google search engine with queries that are reformulations of the original definitions obtained by enriching the morphological forms of the keywords (e.g., by varying number and gender for nouns, or the tense for verbs), by adding synonyms and hypernyms from WordNet, in order to make the querying process more effective. The text in the retrieved pages is analyzed by NLP techniques, and a classifier chooses the most probable part-of-speech depending on the definition in order to reduce the number of candidate words. Finally, a list of the best words for the definition is given and they are matched against the letter constraints given by the grid. WebCrow achieves 68.8% of correct words and 79.9% of correct letters, showing the potential of the Web as a resource for complex language games.

Another interesting language game is the Guillotine, a game broadcast by the Italian National TV company. It involves a single player, who is given a set of five words (clues), each linked in some way to a specific word that represents the unique solution of the game. Words are unrelated to each other, but each of them is strongly related to the word representing the solution. For example, given the five words *sin, Newton, doctor, pie, New York*, the solution is *apple* because: the apple is the symbol of original *sin* in Christian theology; *Newton* discovered the gravity by means of an apple; "an apple a day keeps the *doctor* away" is a famous proverb; the apple *pie* is a fruit pie, and *New York* city is also called "the big apple". In [Semeraro et al., 2012], the authors present OTTHO (On the Tip of my THOught), an artificial player for the Guillotine game. The idea behind OTTHO is to define a *knowledge infusion* process which analyzes unstructured information stored in open knowledge sources on the Web to create a memory of linguistic competencies and world facts that can be effectively exploited by the system for a deeper understanding of the information it deals with. The knowledge infusion process adopts NLP techniques to build a knowledge base and, similarly to the approach described in this chapter, extracts information mainly from Wikipedia. A reasoning mechanism based on a spreading activation algorithm is adopted to retrieve the most appropriate pieces of knowledge useful to find possible solutions.

An approach to implement an artificial player for the "Who Wants to Be a Millionaire?" game has already been proposed [Lam et al., 2003]. The authors exploit the huge amount of knowledge in the Web and use NLP techniques to reformulate the questions in order to create different queries. The queries are then sent to Google search engine and the number of results is used as a ranking mechanism which exploits the redundancy of the information sources [Lam et al., 2003]. A decision making module, that combines results in the spirit

of ensemble learning using an adaptive weighting scheme, tries to maximize the earned amount of money with respect to the risk of answering. The system reaches an accuracy of 75%, showing how unstructured data can be useful for this kind of task, but it fails when questions require common sense reasoning and access to structured information. The main differences with respect to our work are that we adopt selected sources of information available on the Web, such as Wikipedia and DBpedia, rather than the whole Web, in an attempt to improve reliability of the answers; moreover, we adopt a QA framework instead of a search engine in order to improve the process of selecting the most reliable passages.

In February 2011 the IBM Watson supercomputer, adopting technology from the DeepQA project [Ferrucci et al., 2010], has beaten two champions of the Jeopardy! TV quiz. In Jeopardy! the player is given a question but expressed as answer, and has to find the answer which must be expressed as a question. Watson applies several NLP, Information Retrieval (IR) and Machine Learning (ML) techniques focusing on open-domain QA, by answering questions without domain constraints. Watson analyzed 200 millions content elements, both structured and unstructured, including the full text of Wikipedia. The steps of the Watson answering process can be summarized as follows: 1) it acquires knowledge from different data sources, namely encyclopedias, dictionaries, thesauri, journal chapters, databases, taxonomies and ontologies; 2) the input of the problem is treated as a question, then it is analyzed using NLP algorithms and lastly it is classified; 3) candidate answers are generated from the previously acquired knowledge; 4) candidate answers that do not pass a threshold are filtered out. Several scoring algorithms are used to rank the candidate answers in order to give evidence of their quality. Lastly, the scoring criteria are combined to select the final candidate answer. Similarly to the approach implemented in Watson, we adopt QA techniques for solving the WWBM game, and we use the same process of using different scoring criteria, which are eventually combined to return the best candidate answer.

7.3.2 Question Answering for Machine Reading and Answer Validation

QA systems generally deal with simple questions that require almost no inference to find the correct answers, since no real understanding of documents is performed. This historically led to QA architectures based on Information Retrieval techniques, in which the final answers are obtained after focusing on selected portions of retrieved documents and matching sentence fragments or sentence parse trees. Other systems perform a deeper analysis of texts, to solve tasks that involve some kind of reasoning. For example, in the Machine Reading task [Peñas et al., 2013], the goal is to answer questions that require a deep

knowledge of individual short texts and in which systems are required to choose one answer, by analyzing the corresponding test document in conjunction with background text collections. Other complex tasks include Recognizing Textual Entailment (RTE) [Dagan et al., 2005][1] and the Answer Validation Exercise (AVE) [Rodrigo et al., 2008][2]. In RTE, a system must decide whether the meaning of a text T entails the meaning of a different text H, i.e. the hypothesis. Differently, AVE consists in deciding whether an *answer* to a *question* is correct or not according to a *given text*. Systems receive a set of triplets (question, answer and supporting text) and must return a value for each triplet, which can be: VALIDATED, i.e. the answer is correct and supported, although not selected, SELECTED, i.e. the answer is validated and chosen as the output, and REJECTED, i.e. the answer is incorrect or there is no enough evidence of its correctness. In order to solve these tasks, several techniques were adopted, mostly based on the use of lexical processing, syntactic processing, and Named Entities [Rodrigo et al., 2007].

In [Breck et al., 2000] the answer validation is carried out by computing the overlap between the system response to a question and the stemmed content words of a human-generated answer key. The intuition behind this strategy is that a good answer is expected to contain certain keywords, but the exact phrasing does not matter. This is the reason why the authors used stopwords removal and stemming. In [Magnini et al., 2002], the answer validation is based on the intuition that the knowledge which connects an answer to a question can be estimated by exploiting the redundancy of the Web information. More specifically, the hypothesis is that the number of documents retrieved from the Web in which the question and the answer co-occur is a good indicator of the validity of the answer.

More complex strategies propose solutions to answer validation based on a lightweight process of abduction starting from paragraphs of text where the answers can be found [Harabagiu and Maiorano, 1999], or based on the use of semantics. For example, Castillo [Castillo, 2008] builds a model using Support Vector Machines to define whether the implication holds, using a set of lexical and semantic measures to compute the similarity between (hypothesis, text) pairs. Features are based on: 1) the overlap between text and hypothesis (computed by taking into account single words or stems, bigrams and trigrams); 2) the cosine similarity and the Levenshtein distance between the text and hypothesis; 3) the semantic similarity using WordNet. Glöckner [2008] also used shallow feature extraction (like lexical overlap) to validate answers. A local score is then computed, and aggregation is used to determine a combined score for each answer which captures the joint evidence of all snippets supporting the answer.

[1] http://pascallin.ecs.soton.ac.uk/Challenges/RTE/

[2] http://nlp.uned.es/clef-qa/ave/

Ahn et al. [2005] exploited semantic representations and inference techniques in the process of answer extraction from selected passages. Candidate answers are extracted using a "relaxed" unification method that takes advantage of Prolog unification, which allows to assign high scores to perfect matches between terms of the question and passage, and low scores for less perfect matches obtained by relaxed unification. Less perfect matches are granted for different semantic types, predicates with different argument order, or terms with symbols that are semantically related (hypernymy) according to WordNet.

We were inspired by those previous approaches to build the artificial player for the WWBM game, and we borrowed most of the techniques adopted in the AVE task, such as techniques based on lexical processing and computation of an approximate match between passages of text and candidate answers (Section 7.6).

7.3.3 Question Answering over Linked Data

The rapid growth of semantic information published on the Web, in particular through the linked data initiative [Bizer, 2009], poses new challenges when supporting users to query these large amounts of heterogeneous and structured semantic data using natural language interfaces. This led to the rise of ontology-based QA, a new paradigm able to exploit the expressive power of ontologies and to go beyond the representation of user information needs as keyword-based queries.

FREyA [Damljanovic et al., 2011] allows users to enter queries in any form. In a first step, it generates a syntactic parse tree in order to identify the answer type. The processing then starts with a lookup, annotating query terms with ontology concepts using the ontology-based gazetteer OntoRoot. Next, on the basis of the ontological mappings, triples are produced and finally combined to generate a SPARQL query. In a similar way, PowerAqua [Lopez et al., 2012] transforms the query into a set of triples ⟨SUBJECT, PROPERTY, OBJECT⟩ by means of linguistic processing, and maps the triples to suitable semantic resources in various ontologies that are likely to describe the query terms. Given these semantic resources, a set of ontology triples that jointly cover the query is derived and combined into a complete answer, by merging and ranking the various interpretations produced in different ontologies.

QAKiS [Cabrio et al., 2012] is a QA system over DBpedia that focuses on bridging the gap between natural language expressions and labels of ontology concepts by means of the WikiFramework repository. This repository has been built by automatically extracting relational patterns from Wikipedia free text that specify possible lexicalizations of properties in DBpedia. For example, one of the natural language patterns that express the relation BIRTHDATE is *"was born on"*. The approach of using a pattern repository represents a promis-

ing solution for bridging the lexical gap between natural language expressions and ontology labels, which we also used in our work (Section 7.5.1). Similarly to QAKiS, SemSek [Aggarwal and Buitelaar, 2012] also focuses on matching natural language expressions to ontology concepts. This relies on three steps: linguistic analysis, query annotation, and semantic similarity. The query annotation mainly looks for the entities and classes in a DBpedia index that match the expressions occurring in the natural language question. This process is guided by the syntactic parse tree provided by the linguistic analysis. Starting from the most plausible identified resources and classes, SemSek retrieves an ordered list of terms following the dependency tree. In order to match these terms to DBpedia concepts, SemSek uses two semantic similarity measures, one based on Explicit Semantic Analysis [Gabrilovich and Markovitch, 2007], and one based on WordNet.

In [Lopez et al., 2013], the evaluation of these systems in the context of the challenges for question answering systems over linked data (QALD) is presented. Results are encouraging and show that QA over linked data can deliver answers to quite complex information needs expressed in natural language using heterogeneous semantic data, even though the task of mapping natural language to formal queries is not trivial and still has several problems [Cimiano and Minock, 2009].

In Berant et al. [2013] the authors propose a semantic parser that maps questions to answers via latent logical forms. They first generate a number of logical forms exploiting a lexicon that maps logical predicates to phrases built from a knowledge base and a large text corpus. They then calculate the probability of each candidate logical form with a log.linear model that exploits lexical and logical features, trained on question-answer pairs. This approach differs from our because we rely mainly on lexical features for retrieving passages rather than mapping questions to their logical form. At the same time we try to map questions to DBpedia triples exploiting predicate lexicalization, which has some commonalities with their lexicon construction step.

7.4 Artificial Player Architecture

The architecture of the artificial player for the WWBM game is presented in Figure 7.2 and consists of four modules:

- Game Manager: manages the user interface, selects a question for each level of the game, and logs the information about each game for the different players;
- Question Answering: leverages Wikipedia and DBpedia to retrieve and rank the most relevant passages of text useful to identify the correct

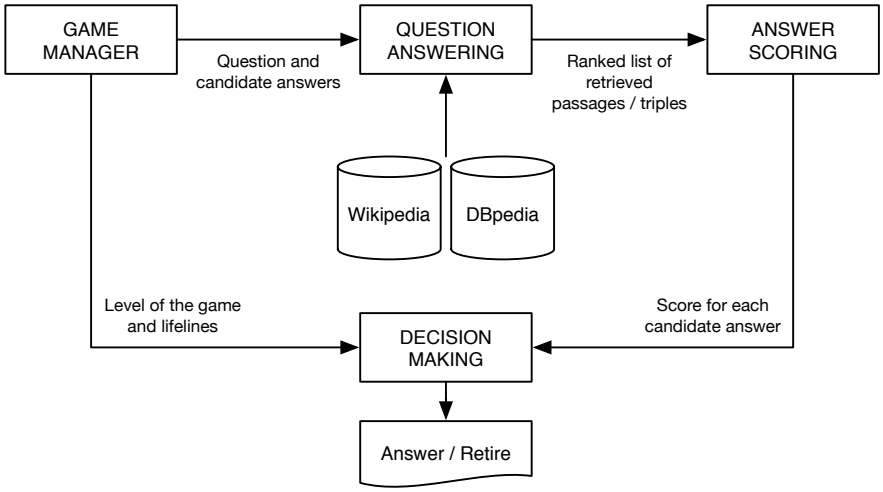


Figure 7.2: Artificial Player architecture.

answer to a question. More details are provided in Section 7.5;

- Answer Scoring: leverages the list of text passages extracted from the Question Answering module and adopts several criteria to assign a score to each of the four possible answers to a question. A detailed description is provided in Section 7.6;

- Decision Making: takes the final decision whether answering to a question or retiring from the game by considering the current level of the game, the available lifelines, and the score computed for each possible answer by the Answer Scoring. A detailed description is provided in Section 7.7.

According to the current level of difficulty of the game, the Game Manager selects a question along with the four possible answers, which are then passed to the Question Answering module. This module exploits the knowledge contained in Wikipedia and DBpedia to select a ranked list of passages of text that likely contain the correct answer to the question. These passages are processed by the Answer Scoring, which implements a set of heuristics to come up with a score for each possible answer to the question. Finally, the Decision Making module decides to provide a specific answer or to retire from the game, by taking into account the scores of the candidate answers, the available lifelines and the current level of the game.

In order to better explain the whole process of answer selection, we will use the following running example throughout the chapter. Let us consider the question in Figure 7.1: *Who directed Blade Runner?*, whose correct answer is B) Ridley Scott. Let us also consider the ranked list of text passages provided

by the Question Answering module in Table 7.1. This example will be used throughout the chapter.

Title (t_i)	**Passage** (p_i)	**Score** (w_i)
Ridley Scott	Sir Ridley Scott (born 30 November 1937) is an English film director and producer. Following his commercial breakthrough with Alien (1979), his best-known works are the sci-fi classic Blade Runner (1982) and the best picture Oscar-winner Gladiator (2000).	5.32
Blade Runner	Blade Runner is a 1982 American dystopian science fiction action film directed by Ridley Scott and starring Harrison Ford, Rutger Hauer, and Sean Young. The screenplay, written by Hampton Fancher and David Peoples, is loosely based on the novel Do Androids Dream of Electric Sheep? by Philip K. Dick.	5.1
Blade Runner	Director Ridley Scott and the film's producers "spent months" meeting and discussing the role with Dustin Hoffman, who eventually departed over differences in vision. Harrison Ford was ultimately chosen for several reasons.	5
Blade Runner	The screenplay by Hampton Fancher was optioned in 1977. Producer Michael Deeley became interested in Fancher's draft and convinced director Ridley Scott to film it.	4.9
Blade Runner	Interest in adapting Philip K. Dick's novel Do Androids Dream of Electric Sheep? developed shortly after its 1968 publication. Director Martin Scorsese was interested in filming the novel, but never optioned it.	1.2

Table 7.1: List of passages returned by the Question Answering module for the question *"Who directed Blade Runner?"*.

Each passage contains the title of the Wikipedia page from which it was extracted, and the score computed by the Question Answering module. More formally, given $\langle q, (A, B, C, D)\rangle$, where q is the question and (A, B, C, D) are the four possible answers, the Question Answering module returns a list of results $R_q = \{\langle t_1, p_1, w_1\rangle, \ldots, \langle t_{|R_q|}, p_{|R_q|}, w_{|R_q|}\rangle\}$ where the triple $\langle t_i, p_i, w_i\rangle$ corresponds to the title t_i of the Wikipedia page containing the passage p_i, and w_i is the score indicating the relevance of that passage with respect to the ques-

tion q. The list R_q is empty if the Question Answering module is not able to find passages relevant to the question. This may happen when:

- the information is not contained in any of the adopted knowledge sources, i.e. Wikipedia or DBpedia;
- the information is contained in one of the adopted knowledge sources, but it is not possible to find a match (textual or semantic) between the question and the passages containing the answer. For example, even though the answer to the question *When was Leonardo da Vinci born?* is contained in Wikipedia, it is difficult to find it, since it is reported as "Leonardo da Vinci (April 15, 1452 - May 2, 1519)". We address this problem by implementing a strategy based on the use of DBpedia (Section 7.5.1);
- the question falls into one of the categories which remain unanswerable by our system (Section 7.8.1).

7.5 Question Answering

We configured the QA system described in Chapter 2 to work on Wikipedia. We adopted all the NLP modules that where available for both English and Italian , we used only the BM25 searcher with terms and lemmas because it was the best performing one and adopted just the Overlap, Density, Other and Distributional Semantics feature extractors, combined with the CombSum strategy.

The configuration of the distributional semantics features is the same presented in Section 6.3.1, but the only adopted corpus for building the semantic spaces was Wikipedia itself.

Moreover, we decided to try to exploit also structured information available in the infoboxes of Wikipedia. In order to do so, we implemented a different candidate answer retrieval strategy from the passage retrieval one. We describe it in the next Section 7.5.1.

7.5.1 Using DBpedia as Knowledge Source

The question *When was Leonardo da Vinci born?* shows the difficulty to extract the correct answer, even though the information is contained in Wikipedia. In this case, the date of birth is "April 15, 1452", but it cannot be identified by adopting the classical passage retrieval process implemented by the Question Answering module. In order to manage this kind of questions, we used a specific search engine which leverages the knowledge contained in DBpedia. DBpedia includes structured information embedded in the Wikipedia chapters – the "infobox". Figure 7.3 reports the infobox for Leonardo da Vinci, which

contains a good deal of useful information, such as birth date, death date, nationality, etc. DBpedia represents resources, properties as well as relations between resources using RDF triples, which contain three components: subject, predicate, and object. For example, the RDF triple that represents the birth date of Leonardo da Vinci is:

⟨ `http://dbpedia.org/resource/Leonardo_da_Vinci`,
`dbpedia-owl:birthDate`,
`1452-04-15` ⟩

DBpedia allows querying relations and properties of Wikipedia resources, thus the birth date of Leonardo da Vinci can be found by accessing the property `dbpedia-owl:birthDate`.

In order to leverage the knowledge contained in DBpedia, we have manually created a mapping between the 50 most frequent DBpedia properties and different lexicalizations, in question form, asking for those specific properties. For example, the property `dbpedia-owl:birthDate` is mapped to the questions *When was he born?*, or *What is the date of birth?*, and other similar wordings. In this way we have obtained two datasets containing 347 questions for Italian and 312 for English, where each question is tagged with the corresponding DBpedia property. Each dataset is used to train a Rocchio classifier [Rocchio, 1971] that, given a question, is able to predict the DBpedia property it is mapped to among the 50 most frequent properties. The features used to train each classifier are all the words occurring in the questions. Stopwords are not removed, since words such as *When*, *How*, *Where* are useful hints to guide the classifier to the correct classification.

In order to retrieve relevant information from DBpedia we queried an additional search engine containing documents which are the lexicalization of RDF triples with the same subject in the form: ⟨*label of the subject*, *label of the predicate*, *label of the object*⟩ (value of the object, in case of literals). The lexicalization for the previous example is: ⟨Leonardo da Vinci, date of birth, 1452-04-15⟩. Only triples related to the 50 selected properties are lexicalized in that way. Each document has an additional field reporting the DBpedia properties it contains, such as `dbpedia-owl:birthDate`. When a query (question) is sent to the DBpedia search engine, it is first classified using the Rocchio classifier in order to identify the property it refers to; then, the selected property is added to the query, along with the named entities (if any) occurring in the question. The query is submitted to the search engine, which retrieves the set of documents relevant to the query. Starting from the documents, the system extracts the corresponding list of passages (the RDF triples), which are scored using the feature extractors described in Section 7.5, and an additional DBpedia property filter. This filter scores the triples containing the property returned by the classifier with its confidence, and the triples not containing it with 0. It is worth

Leonardo da Vinci

Portrait of Leonardo by Melzi

Born	Leonardo di ser Piero da Vinci April 15, 1452 Vinci, Republic of Florence (present-day Italy)
Died	May 2, 1519 (aged 67) Amboise, Kingdom of France
Known for	Diverse fields of the arts and sciences
Notable work(s)	*Mona Lisa* *The Last Supper* *The Vitruvian Man* *Lady with an Ermine*
Style	High Renaissance
Signature	

Figure 7.3: Leonardo da Vinci infobox.

to note that each question is always submitted to both search engines working on Wikipedia and DBpedia, in order to retrieve results from both knowledge sources.

7.6 Answer Scoring

The main goal of the Answer Scoring module is to assign a score to each of the four possible answers to a question. Similarly to the approaches used in the context of the Answer Validation Exercise [Rodrigo et al., 2008], we adopted five criteria based on the analysis of the passages returned by the Question An-

swering module. Each criterion returns a score for each possible answer, which is normalized using the sum of the scores of the four possible answers. More formally, given $\langle q, (A, B, C, D)\rangle$, each criterion computes $\langle q, (c_A, c_B, c_C, c_D)\rangle$, where c_X is the score assigned to the candidate answer X. In the following, each criterion is described by taking into account our running example.

Title Levenshtein (TL) criterion. This criterion computes the Levenshtein distance between a candidate answer X and the title t_i of the Wikipedia page returned by the Question Answering module. The Levenshtein distance measures the difference between two strings, and is defined as the minimum number of single-character edits (i.e. insertion, deletion, substitution) required to change one string into the other. As the Levenshtein distance is a distance measure, rather than a similarity measure, we compute:

$$\frac{max(len(X), len(t_i)) - lev(X, t_i)}{max(len(X), len(t_i))}$$

where $len(\cdot)$ is the function computing the length of the string, and $lev(X, t_i)$ is the Levenshtein distance between X and t_i, used as a normalization factor (to have scores in the $[0, 1]$ interval). In our running example, taking into account just the first passage returned by the Question Answering module (Table 7.1), the answer B) Ridley Scott occurs in the title of the page containing the passage, so it gets the maximum score of 1, while the answer A) Harrison Ford gets a score equal to $\frac{13-12}{13} = 0.077$, the answer C) Philip Dick gets a score equal to $\frac{12-10}{12} = 0.167$, and the answer D) James Cameron gets a score equal to $\frac{13-12}{13} = 0.077$. The normalized score of each answer is:

$$c_A = \frac{0.077}{0.077+1+0.167+0.077} = 0.058$$

$$c_B = \frac{1}{0.077+1+0.167+0.077} = 0.757$$

$$c_C = \frac{0.167}{0.077+1+0.167+0.077} = 0.126$$

$$c_D = \frac{0.077}{0.077+1+0.167+0.077} = 0.058$$

Longest Common Subsequence (LCS) criterion. This criterion computes the Longest Common Subsequence between a candidate answer X and a passage of text p_i returned by the Question Answering module, or just the title t_i of the Wikipedia page the passage p_i is extracted from. In our running example, taking into account only the second passage, the answer A) Harrison Ford gets a score equal to 13, the answer B) Ridley Scott gets a score equal to 12, the answer C) Philip Dick gets a score equal to 11, and the answer D) James Cameron gets a score equal to 0 since it does not occur in the passage. The normalized score of each answer is:

$$c_A = \frac{13}{13+12+11+0} = 0.361$$

$$c_B = \frac{12}{13+12+11+0} = 0.333$$

$$c_C = \frac{11}{13+12+11+0} = 0.305$$

$$c_D = \frac{0}{13+12+11+0} = 0$$

Overlap criterion. This criterion computes the Jaccard index between the set of terms in a candidate answer X and the set of terms in a passage of text p_i returned by the Question Answering module. The Jaccard index measures the similarity between sets, and is defined as the size of the intersection divided by the size of the union of the sets. In our running example, taking into account only the second passage, answers A), B) and C) get a score of $\frac{2}{49} = 0.041$, while the answer D) gets a score equal to 0. The normalized score of each answer is:

$$c_A = c_B = c_C = \frac{0.041}{0.041+0.041+0.041+0} = 0.333$$

$$c_D = \frac{0}{0.041+0.041+0.041+0} = 0$$

Exact Substring (ES) criterion. This criterion computes the length in characters of the longest common substring between a candidate answer X and a passage of text p_i returned by the Question Answering module, normalized using the length of the candidate answer. In our running example, taking into account only the second passage returned by the Question Answering module, the answer A) Harrison Ford gets a score of $\frac{13}{13} = 1$, the answer B) Ridley Scott gets a score of $\frac{12}{12} = 1$, the answer C) Philip Dick gets a score of $\frac{6}{11} = 0.55$, and the answer D) James Cameron gets a score equal to 0. The normalized score of each answer is:

$$c_A = \frac{1}{1+1+0.55+0} = 0.392$$

$$c_B = \frac{1}{1+1+0.55+0} = 0.392$$

$$c_C = \frac{0.55}{1+1+0.55+0} = 0.215$$

$$c_D = \frac{0}{1+1+0.55+0} = 0$$

Density criterion. This criterion computes the density of the terms in a candidate answer X inside a passage of text p_i returned by the Question Answering module, using the minimal overlapping span method [Monz, 2004] modified as described in Section 2.3.3 and Section 6.3.1. In our running example, taking into account only the second passage returned by the Question Answering module, the answers A) and B) get a score equal to 1, the answer C) gets a score equal to $\frac{2}{3} = 0.66$ (as the passage reports the full name Philip K. Dick, adding an extra token between the two tokens of the candidate answer), and the answer D) gets 0. The normalized score of each answer is:

$$c_A = \frac{1}{1+1+0.66+0} = 0.376$$

$c_B = \frac{1}{1+1+0.66+0} = 0.376$

$c_C = \frac{0.66}{1+1+0.66+0} = 0.248$

$c_D = \frac{0}{1+1+0.66+0} = 0$

Each criterion has some parameters that can be set:

1. *number of processed passages*: in this case the score of each answer is computed for each of the top-n passages returned by the Question Answering module, and the final score is the average of those values;

2. *use of the weight w_i of the passages* returned by the Question Answering module: in this case the average computed at the previous point is weighted using the score w_i of each passage. This strategy allows to assign higher weights to passages deemed as more relevant to the question by the Question Answering module;

3. *level of linguistic analysis* to adopt for processing the passages returned by the Question Answering module. Passages are represented by using keywords, lemmas, or stems, with or without stopword removal;

4. *use of the question expansion*: the system asks four different questions obtained by concatenating the original one with each of the four possible candidate answers. In our running example, the artificial player queries the Question Answering module using the following questions *Who directed Blade Runner? Harrison Ford*, *Who directed Blade Runner? Ridley Scott*, *Who directed Blade Runner? Philip Dick* and *Who directed Blade Runner? James Cameron*. If a criterion adopts the question expansion, it uses four different sets of passages (one for each question), instead of using the same set of passages.

Besides the above mentioned criteria for Answer Scoring, the distributional filter (Section 7.5) could be also adopted for comparing answers against the retrieved passages. We finally decided to avoid using that filter because, after the selection of the most relevant passages to a specific question, we expect that the candidate answer is already contained in one of the passages, and the answer scoring criteria based on pure lexical comparison are sufficient to properly select and score it. Moreover, the answers usually contain one or two words, and most of them are named entities, numbers, or dates, which could be hardly identified by a distributional filter.

It is worth to mention how we managed a specific type of questions, i.e. those formulated in negative form, such as *Quale di questi super-criminali non è uno storico nemico di Batman?* (in English *Which of these villains is not a historical enemy of Batman?*). Let us suppose that the candidate answers

are *Dottor Destino* (*Doctor Doom*), *Mister Freeze*, *Pinguino* (*Penguin*), and *Joker*. The Question Answering module returned passages containing *Joker*, *Pinguino*, and *Mister Freeze*, that would be correct if the question was raised in positive form. In fact, when the question is raised in negative form, the top-ranked passages often contain the candidate answers that are actually those to be discarded. For this reason, when the system detects a negative question, the strategy for scoring the candidate answers is reversed, meaning that those with the highest score become the less plausible and vice versa. The process of detecting questions raised in negative form leverages several heuristics based on the use of regular expressions.

7.7 Decision Making

The Decision Making module is responsible for the decision about answering to a specific question, retiring from the game or using one of the available lifelines. The decision strategy evaluates the uncertainty of the information provided by the Question Answering module and the Answer Scoring, the available lifelines, and the level of the question in order to take the final decision. The decision making strategy we devised encapsulates two heuristics to manage the following situations of uncertainty:

1. the maximum score computed by the Answer Scoring for the four candidate answers is very low. This means that either the quality of the passages retrieved by the Question Answering module may be low, hence those passages are not useful to find the correct answer to the question, or the criteria adopted by the Answer Scoring for assigning a score to the candidate answers are not satisfactory;

2. the difference between the score of the best candidate answer and the second best candidate answer is very small. This means that the artificial player is not able to provide clear evidence of the most likely candidate answer between the two candidate answers.

In a situation in which the artificial player has enough *confidence* in one of the candidate answers, it can answer the question without using any lifeline. In a situation of uncertainty, the decision making algorithm can take one of the following decisions: 1) to retire from the game; 2) to use one or more available lifelines; 3) to continue to play by providing a random answer. The random strategy can be also useful in some situations, in particular at specific levels of the game (6th or 11th question) when a wrong answer causes no loss in the earned amount.

The decision making algorithm implements a simple strategy for managing the lifelines, meaning that the order of their usage is independent of the level of

the current question. The system leverages first the *Poll the Audience* lifeline and, if it does not return a candidate answer with a good confidence, the *Phone a Friend* is explored, and finally *50:50*. Other authors faced the problem of defining a more dynamic decision making strategy for the WWBM game. In [Lam et al., 2003], the decision making module constructs a decision tree that encodes the probabilities and utilities at each potential future state of the game. The tree consists of both decision forks for choosing whether to answer the question, to use a lifeline, or to walk away, and chance forks to encode the uncertainty of correctly answering the questions. The best choice is obtained with the action that maximizes the expected utility. The strategy is able to parameterize the risk of the artificial player, which can exhibit a risk averse behavior or a more risk neutral one. Probabilities are assigned to the nodes of the tree based on historical past performance on a sample of questions from the associated difficulty level. A different strategy based on dynamic programming is adopted in [Perea and Puerto, 2007] to analyze two different objectives: 1) to maximize the expected reward, 2) to maximize the probability of reaching a given question. An analysis of the results presented in that work allowed us to define the order in which our decision making module should check and use the available lifelines. More specifically, we found out that *Poll the Audience* must be checked before the others, while no specific indications are provided for *50:50* and *Phone a Friend*.

The functions used by our decision making algorithm (Algorithm 2) are the following:

- Best(Answers) and SecondBest(Answers) return the best and second best candidate answer to a question q, respectively;
- CanUse(lifeline) returns *true* if that specific *lifeline* has not yet been used in the current game, otherwise it returns $false$;
- Use(lifeline) returns a new set of answers together with their scores obtained by adopting that specific *lifeline*;
- CanRisk() returns *true* if the current question allows the player to provide a wrong answer without losing the earned money (6th or 11th question), otherwise it returns $false$;
- Random(Answers) allows the artificial player to provide a random answer to a question;
- Retire() allows the artificial player to retire and win the earned money.

The artificial player uses no lifelines if it has enough confidence in one of the four answers (`if` statement at step 4). In this case the provided answer is the one with the highest score (step 33). If the artificial player is in a situation

of uncertainty, it explores the available lifelines starting by *Poll the Audience* (steps 5-11). If the player has enough confidence in the answer provided by that lifeline (step 8), it returns that answer (step 9), otherwise it continues to explore the other available lifelines. Steps 12-18 manage the *Phone a Friend* lifeline, which allows returning the answer possibly provided by a friend (step 16). The usage of the *50:50* lifeline is related to the level of the game reached by the player. If the user can risk, meaning that the money would not be lost even providing a wrong answer to the question (step 19), the player uses the *50:50* lifeline (step 20). The player chooses the answer with the highest score (step 23) if that value is higher than a certain threshold, otherwise it randomly returns one of the two remaining answers (step 25). If the player cannot rely on lifelines, or it does not have enough confidence in the answers provided by lifelines, then it returns a randomly chosen answer in case the user can risk (steps 28-30), otherwise it retires from the game (step 31).

Given that the game is played using a board game variant, we implemented a specific strategy to simulate the actual way of using lifelines, as their use in the board game implies the interaction with the other players. *50:50* is simulated in the same way as in the real game, i.e. by removing two wrong answers among the four candidate answers. As in the real game, *Phone a Friend* and *Poll the Audience* lifelines are not always able to return the correct answer. As regards *Phone a Friend*, it is worth to notice that usually the higher the level of the game, the more difficult is to answer to the question. Hence, this lifeline works as follows: it always returns the correct answer when used for levels from 1 to 5; it randomly chooses between two alternatives, i.e. providing the correct answer or not returning the answer at all, when used for levels from 6 to 10; it randomly chooses between three alternatives, i.e. providing the correct answer, not returning the answer at all or returning the wrong answer, when used for levels from 11 to 15. On the other side, *Poll the Audience* is simulated in order to distribute the votes coming from the audience (in percentage) among the candidate answers. Without a real audience, we simulated the distribution of votes using a strategy which takes into account both the current level of the question and a degree of randomness. We first assign a percentage of all votes to the correct answer, representing the percentage of the audience that would know the correct answer, then we randomly distribute the remaining votes among the other candidate answers. The percentage of votes assigned to the correct answer – baseline – is inversely proportional to level of the game, i.e. the more difficult the question, the lower the confidence; then, the baseline percentage fo votes is randomly perturbed according to the level of the game, between the lower and upper bound depicted in Figure 7.4. It is worth noting that the perturbation of the baseline percentage is different for questions whose level is between 1 and 5, 6 and 10, 11 and 15. For example, if the player uses

Algorithm 2 Decision making algorithm

```
1: procedure DECISION MAKING(< q, (c_A, c_B, c_C, c_D) >, lifelines)  ▷
   Decision strategy based on the scores of the four candidate answers for question q, and
   the available lifelines
2:     BestAnswer ← BEST(< q, (c_A, c_B, c_C, c_D) >)
3:     SecondBestAnswer ← SECONDBEST(< q, (c_A, c_B, c_C, c_D) >)
4:     if BestAnswer.score < threshold_1
   or (BestAnswer.score − SecondBestAnswer.score)
   < (BestAnswer.score ∗ threshold_2) then
5:         if CANUSE(Poll the Audience) then
6:             audienceAnswers ← USE(Poll the Audience)
7:             lifelines ← lifelines − {Poll the Audience}
8:             if audienceAnswers.score > threshold_1 then
9:                 RETURN BEST(audienceAnswers)
10:            end if
11:        end if
12:        if CANUSE(Phone a Friend) then
13:            friendAnswer ← USE(Phone a Friend)
14:            lifelines ← lifelines − {Phone a Friend}
15:            if friendAnswer ≠ null then
16:                RETURN friendAnswer
17:            end if
18:        end if
19:        if (CANUSE(50:50) and CANRISK()) then
20:            50 : 50answers ← USE(50:50)
21:            lifelines ← lifelines − {50:50}
22:            if 50 : 50answers.score > threshold_1 then
23:                RETURN BEST(50:50answers)
24:            else
25:                RETURN RANDOM(50:50answers)
26:            end if
27:        end if
28:        if CANRISK() then
29:            RETURN RANDOM(answers)  ▷ No more lifelines but the player can risk
30:        end if
31:        RETIRE()
32:    else
33:        RETURN BestAnswer
34:    end if
35: end procedure
```

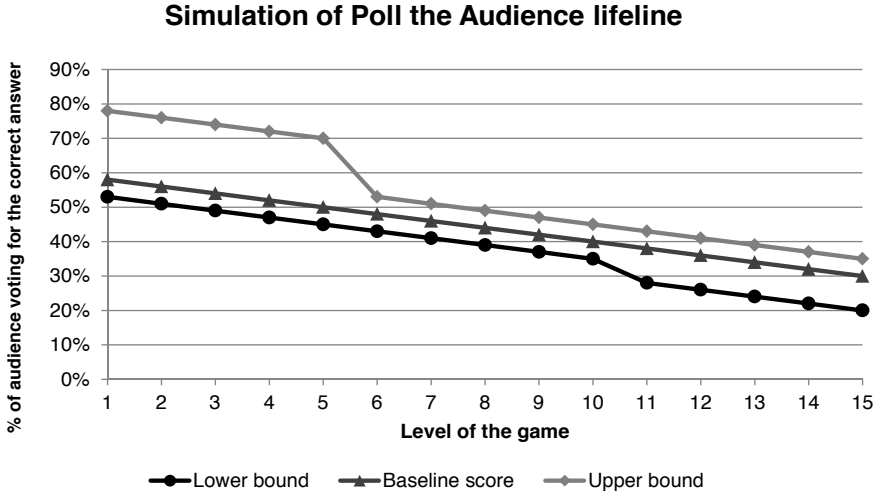


Figure 7.4: Variability of the score assigned to the correct answer when *Poll the Audience* lifeline is used.

Poll the Audience at level 3, the baseline percentage is equal to 54%, and it is randomly perturbed between 49% and 74%; the remaining votes (between 51% and 26%) are randomly distributed among the remaining answers; if the lifeline is used at level 14, the baseline percentage is 32% and it is randomly perturbed between 22% and 37%, and the remaining votes (between 78% and 63%) are randomly distributed among the remaining answers.

7.8 Experimental Evaluation

The goal of the evaluation is twofold:

1. to assess the effectiveness of the Question Answering and Answer Scoring modules to answer to questions of the game, and compare the results with those obtained by human players. The experiment is discussed in Section 7.8.1;

2. to evaluate the accuracy of the artificial player to play the game, taking also into account the strategy implemented by the Decision Making module, and to compare the results with those obtained by human players. The experiment is discussed in Section 7.8.2.

We used two datasets: one containing questions from the WWBM official Italian board game, and one containing questions from the English board game[3].

[3]Datasets available upon request

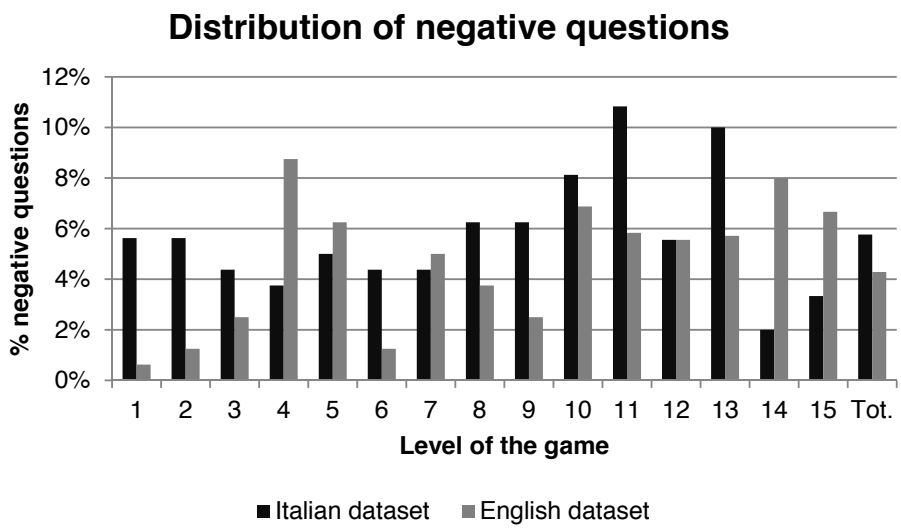


Figure 7.5: Distribution of negative questions per level of the game.

Both datasets contain 1,960 questions, subdivided in 15 groups, one for each level of the game. The first 10 levels contain 160 questions each, while levels 11, 12, 13, 14 and 15 contain 120, 90, 70, 50 and 30 questions, respectively. 113 questions for Italian, and 84 for English are in negative form. The distribution of negative questions per level of the game is reported in Figure 7.5.

While previous results to compare with are available for the English version of the game (even though not on the same dataset), we are not aware of previous results for Italian, i.e. we do not have any baseline to compare with. The metric adopted for the evaluation is the *accuracy*, i.e. the proportion of correctly answered questions, computed as the ratio between the number of correct answers (n_c) and the total number of questions (n): $accuracy = \frac{n_c}{n}$. The significance of the results is assessed using the McNemar's test with Bonferroni correction.

7.8.1 Experiment 1: Evaluation of the Performance of QA and Answer Scoring

The goal of this experiment is to assess the accuracy of the five criteria for answer scoring, properly configured using the parameters described in Section 7.6, namely:

- number of processed passages: we tested 10 configurations, by using the top-n, $n = 1, 2, 3, 4, 5, 10, 15, 20, 25, 30$ passages returned by the QA module;
- use of the score w_i for the passages returned by the QA module: we tested 2 configurations, *using* and *not using* the weight for each passage returned

by the QA module;

- level of linguistic analysis adopted to process the passages returned by the QA module: we tested 3 configurations, which represent passages using *keywords*, *stems*, or *lemmas*. We also run 2 further configurations, *using* and *not using stopwords removal*.

- use of question expansion: we tested 2 configurations, *with* and *without question expansion*.

Overall, we have a set of 1,200 different configurations used to assign a score to each of the four possible answers to a question of the game: starting from these scores, the answer with the highest one is selected. The Question Answering system was configured as described in Section 7.5.

Table 7.2 and Table 7.3 report the accuracy of the best 15 configurations, averaged over the whole set of 1,960 questions of the Italian and English datasets.

Rank	**Criterion**	**P**	**Lex**	**S**	**SW**	**QE**	**Accuracy**
1	Overlap	25	ST	Y	Y	N	64.29%
2	Overlap	25	LEM	Y	Y	N	64.29%
3	Density	3	KWD	Y	N	Y	64.03%
4	Density	30	ST	Y	Y	N	64.03%
5	Density	30	LEM	Y	Y	N	64.03%
6	Overlap	20	ST	Y	Y	N	63.78%
7	Overlap	20	LEM	Y	Y	N	63.78%
8	Overlap	30	ST	Y	Y	N	63.78%
9	Overlap	30	LEM	Y	Y	N	63.78%
10	Density	20	ST	Y	Y	N	63.27%
11	Density	20	LEM	Y	Y	N	63.27%
12	Density	25	KWD	Y	Y	N	63.01%
13	Overlap	15	ST	Y	Y	N	62.76%
14	Overlap	15	LEM	Y	Y	N	62.76%
15	Overlap	20	ST	N	Y	N	62.76%

Table 7.2: Performance of the top-15 configurations (averaged over all the questions) on the Italian dataset.

Acronyms: **P** # of passages, **Lex** Lexicalization, **KWD** Keywords, **LEM** Lemmas, **ST** Stems, **S** Use of the score of the passage, **SW** use of stopword removal, **QE** Use of the question expansion.

It is worth to note that *Overlap* and *Density* are the best performing criteria for both the Italian and the English datasets (the top-85 best configurations are

Rank	Criterion	P	Lex	S	SW	QE	Accuracy
1	Overlap	25	LEM	Y	Y	N	59.47%
2	Overlap	25	ST	Y	Y	N	59.38%
3	Density	3	KWD	Y	N	Y	59.26%
4	Overlap	20	ST	Y	Y	N	59.22%
5	Density	30	ST	Y	Y	N	59.08%
6	Density	30	LEM	Y	Y	N	59.08%
7	Overlap	30	ST	Y	Y	N	58.99%
8	Density	20	ST	Y	Y	N	58.84%
9	Overlap	15	LEM	Y	Y	N	58.72%
10	Density	25	KWD	Y	Y	N	58.37%
11	Overlap	30	LEM	Y	Y	N	58.35%
12	Density	20	LEM	Y	Y	N	58.21%
13	Overlap	20	LEM	Y	Y	N	58.14%
14	Overlap	20	ST	N	Y	N	57.99%
15	Overlap	15	ST	Y	Y	N	57.97%

Table 7.3: Performance of the top-15 configurations (averaged over all the questions) on the English dataset.

Acronyms: **P** # of passages, **Lex** Lexicalization, **KWD** Keywords, **LEM** Lemmas, **ST** Stems, **S** Use of the score of the passage, **SW** use of stopword removal, **QE** Use of the question expansion.

obtained using those criteria). The analysis of the results unveils the usefulness of taking into account a considerable number of passages, the usefulness of leveraging the score of the passages returned by the QA module, and the effectiveness of the process of stopwords removal. Finally, the question expansion process seems to negatively affect the accuracy of the system. The worst criterion is *Title Levenshtein* for both Italian and English: indeed, the worst 100 results are obtained by that criterion.

In order to have a clear picture of the accuracy of the whole set of configurations, Table 7.4 and Table 7.5 report, for each criterion, its best and worst configuration.

We observe that the best and worst configurations of each criterion for Italian and English are pretty much the same, even though the accuracy obtained on the English dataset is 5% lower than that obtained for Italian, on average. Statistical tests show that there is no significan difference among the best configuration of the overlap criterion and the best configuration of the density criterion, but both their improvements are statistically significant ($p < 0.01$) with respect to the best configurations of other criteria, regardless of the language.

Starting from the accuracy of the single configurations, we tried to combine

Rank	Criterion	P	Lex	S	SW	QE	Accuracy
BEST	Overlap	25	ST	Y	Y	N	64.29%
WORST	Overlap	1	KWD	Y	N	N	42.60%
BEST	Density	3	KWD	Y	N	Y	64.03%
WORST	Density	1	LEM	N	N	N	43.37%
BEST	ES	30	KWD	Y	Y	N	59.18%
WORST	ES	1	KWD	Y	N	N	42.09%
BEST	LCS	3	KWD	Y	N	Y	57.14%
WORST	LCS	1	LEM	Y	Y	N	41.07%
BEST	TL	1	KWD	Y	Y	Y	40.05%
WORST	TL	1	LEM	Y	N	N	20.15%

Table 7.4: Best and worst performance of each single criterion along with its configuration (averaged over all the questions) for Italian.

Acronyms **P** # of passages, **Lex** lexicalization, **S** Use of the score of the passage, **SW** use of stopword removal, **QE** Use of the query expansion, **ES** Exact Substring, **LCS** Longest Common Subsequence, **TL** Title Levenshtein, **KWD** Keywords, **LEM** Lemmas, **ST** Stems.

Rank	Criterion	P	Lex	S	SW	QE	Accuracy
BEST	Overlap	25	LEM	Y	Y	N	59.47%
WORST	Overlap	1	KWD	Y	N	N	37.74%
BEST	Density	3	KWD	Y	N	Y	59.26%
WORST	Density	1	ST	Y	N	N	38.58%
BEST	ES	30	KWD	Y	Y	N	54.62%
WORST	ES	1	KWD	Y	N	N	37.46%
BEST	LCS	3	KWD	Y	N	Y	52.19%
WORST	LCS	1	LEM	Y	Y	N	36.04%
BEST	TL	1	KWD	Y	Y	Y	35.17%
WORST	TL	1	LEM	N	N	N	15.12%

Table 7.5: Best and worst performance of each single criterion along with its configuration (averaged over all the questions) for English.

Acronyms **P** # of passages, **Lex** lexicalization, **S** Use of the score of the passage, **SW** use of stopword removal, **QE** Use of the query expansion, **ES** Exact Substring, **LCS** Longest Common Subsequence, **TL** Title Levenshtein, **KWD** Keywords, **LEM** Lemmas, **ST** Stems.

them in order to improve the overall accuracy of the Answer Scoring. In [Molino et al., 2013a], we carried out a *greedy* combination: we started with the

best performing configuration and we iterate by greedily adding one configuration at a time by selecting exclusively those that provided an improvement in terms of accuracy.

In this work we propose a combination based on a pointwise learning to rank approach using regression-based algorithms [Liu, 2011], in which question-answer pairs (q, a) are labeled with the relevance judgments of the answer a with respect to the question q. In our setting, the correct answer to a question of the game is labeled with the relevance judgment 1, while the other three incorrect answers are labeled with 0. Each training example is represented using a feature vector consisting of the *level of the question (from 1 to 15)*, and all the *1,200 single scores* obtained by the above mentioned configurations. We opted for using Random Forests (RF) [Breiman, 2001] algorithm, an ensemble learning method, combining different *tree regressors* built using different samples of the training data and random subsets of data features that we already described in Section 4.4. The final result is obtained by averaging the output of the single trees. The use of different data samples from the same distribution and of different feature sets for learning the individual trees prevent overfitting. We adopted the implementation provided by the RankLib library[4].

Questions in each dataset are split into a training set *Tr*, and a test set *Ts*. The methodology adopted for obtaining *Tr* and *Ts* was the stratified 5-fold cross validation, where the stratification process ensures each fold contains the same distribution of questions for the different levels of the game. Given the size of each dataset (1,960 questions), applying 5-fold cross validation means that the dataset is divided into 5 disjoint partitions, each containing 392 questions. The experiment was performed in 5 steps. At each step, 4 partitions were used as training set *Tr* (1,568 questions), whereas the remaining partition was used as test set *Ts*. The steps were repeated until each of the 5 disjoint partitions was used as the *Ts*, and results were averaged over the 5 runs. In order to tune the parameters of the learning to rank (number of trees, learning rate, subsampling rate), we also used a *validation* set, obtained by sampling questions from the training set of each run. We sampled 12.5% of the training set (196 questions), and even in this case we took into account the distribution of the questions among the different levels (stratification). To sum up, at each step, the training set contains 1,372 questions, the validation set contains 196 questions, and the remaining 392 questions are used as test set. The final accuracy obtained by combining *all* the individual configurations through the learning to rank strategy is equal to 79.64% for Italian, and 76.41% for English (averaged over the five runs), which is significantly better ($p << 0.0001$) than the accuracy of the best single configuration (64.29% for Italian, 59.47% for English). The result obtained for English is similar to that achieved in [Lam et al., 2003], in which

[4] http://sourceforge.net/p/lemur/wiki/RankLib/

the system correctly answered about 75% of questions, even though a different and smaller dataset was used. More details about the accuracy for each different level of difficulty of the game is presented and discussed in Section 7.8.1.

Unanswerable Questions and Error Analysis

In order to understand the questions for which the system is not able to provide a correct answer, we classified them in specific categories, and we performed a proper error analysis. The following list contains categories of questions which remain unanswerable by our system:

- *Questions regarding concepts not occurring in Wikipedia / DBpedia:* some of these questions concern astrology, proverbs and sayings, and religion.
- *Questions regarding concepts which are not explicit in Wikipedia / DBpedia:* the system may not be able to provide an answer to some questions, even though the necessary information is contained in the knowledge sources. For example, the question *Quale di questi attori non è figlio d'arte?* (in English *Which of these actors is not an actor son of an actor father?*) would require to match the concept *figlio d'arte* (actor son of an actor father), which is not explicit in Wikipedia / DBpedia.
- *Questions whose answers involve special numbers, periods of time, and mathematical computations:* the system fails to provide an answer when it contains Roman numerals, or it implies the computation of time frames, or some mathematical computations. For example, the candidate answers to the question *In quale secolo fu costruita la prima penna a sfera?* (in English *In what century was the first ballpoint pen built?*) are A) XX B) XVII C) XVIII D) XIX, but the system is not able to find a match with any of the candidate answers; the question *How long was William Harrison in office as the ninth president of USA?* involves the computation of a time frame which the system is not able to carry out; the question *How many degrees are each of the other two angles in an isosceles triangle, if one angle is* 120°*?* requires that the system knows that the total of all angles in any Euclidean triangle would sum to 180°.
- *Questions that require language proficiency:* questions such as *Nella lingua latina, il vocativo plurale è sempre uguale a che cosa?* (in English *In Latin, the vocative plural is always equal to what?*), would require a specific knowledge which is not encoded in Wikipedia / DBpedia.
- *Questions that require making a comparison:* questions such as *Quale tra queste regioni italiane ha la superficie minore?* (in English *Which of these regions of Italy has the lowest surface?*) would require to make a comparison and the selection of the minimum value.

- *Questions that require knowledge of visual properties:* questions such as *Quale di queste squadre di calcio non ha come simbolo un esemplare di lupo?* (*Which of these football teams does not have a wolf as symbol?*) would require the knowledge of visual properties.

The error analysis carried out on the results of the experiments in the previous section unveils that for both Italian and English about 13% of errors are due to the lack of information in Wikipedia / DBpedia, about 57% concerns: 1) questions whose concepts are not explicit in Wikipedia / DBpedia (15%); 2) questions which involve numbers, time and math (13%); 3) questions which require language proficiency (14%); 4) questions which require to make a comparison (7.50%); 5) questions which require knowledge about visual properties (7.50%). The remaining 30% concerns errors due to the wrong scoring of answers or due to other factors, such as the heuristics to recognize and manage negative questions (Section 7.6).

Hence, in order to evaluate how much effective that strategy is, we have evaluated on one hand the accuracy of regular expressions at correctly detecting negative questions, and on the other hand the effectiveness of the reverse scoring method. The accuracy of the heuristics based on the use of regular expressions is $F1 = 89.23\%$ for the Italian dataset, and $F1 = 98.20\%$ for the English one, while the reverse scoring method was evaluated by taking into account the percentage of negative questions for which the reverse scoring strategy was useful (i.e. the answer with the lowest score was correct), harmful (i.e. the answer with the lowest score was wrong) or indifferent (i.e. the correct answer is neither the one with the highest score nor the one with the lowest score): for the Italian dataset percentages are 80%, 9%, and 11%, respectively, while for the English dataset percentages are 96%, 1%, and 3%, respectively. This means that the reverse scoring method is effective in practice.

Finally, in order to have some details about the DBpedia benefit, we have computed the number of questions for which the QA module returned passages coming from DBpedia. 290 questions for the Italian dataset, and 293 for the English one rely on passages coming from DBpedia (besides those coming from Wikipedia), and this means that the answer for those questions could potentially be extracted from DBpedia. For the sake of completeness we have also evaluated the accuracy of the Rocchio classifier described in Section 7.5.1, which allows mapping different lexicalizations of questions asking for a specific property to the corresponding DBpedia property. The accuracy of the classifier, computed using leave-one-out is 85.59% for the English dataset, and 83.57% for the Italian one.

Ablation Tests

To gain insights about the power of the different parameters used to configure each answer scoring criterion, we performed feature selection using *ablation tests*, by removing single features or groups of features.

As regards the ablation of single features, we trained a different model by removing each feature related to each single configuration, and measuring the corresponding predictive accuracy of the learning to rank model (i.e. the process is repeated 1,200 times). The analysis of results unveils that only 160 out of 1,200 times the accuracy of the learned models is lower than that obtained by the best configuration, regardless of the language adopted. The maximum decrease of accuracy is 4.84% for Italian, and 5.00% for English.

It is interesting to note that in each one of those 160 learned models a feature corresponding to a configuration adopting keywords for representing passages is removed. Hence the signal brought by *keywords* is more relevant than those brought by stems and lemmas.

We also performed ablation tests by removing *groups* of features. We defined the following twelve different groups:

- Five groups, each corresponding to a different answer scoring criterion, i.e. each group contains all the features corresponding to all the configurations of each single criterion;
- Three groups, each corresponding to a different level of linguistic analysis adopted for processing the passages of text returned by the QA module;
- Two groups, each containing all the configurations using and not using the question expansion strategy;
- Two groups, each containing all the configurations using a number of passages from 1 to 5, and from 10 to 30, respectively.

Figure 7.6 reports the decrease of accuracy obtained by different models trained by removing each single group of features.

Groups with higher values are better, meaning that those features have a higher impact on the overall performance of the model, since their ablation leads to a higher decrease of accuracy. All the results are statistically significant when compared to the best configuration obtained by the learning to rank strategy ($p << 0.0001$).

As regards the groups corresponding to the answer scoring criteria, the best performance is obtained by LCS, followed by TL, Density, Overlap and ES. A comparison with the performance of the single configurations reported in Table 7.2, Table 7.3, Table 7.4 and Table 7.5, shows that *Overlap* and *Density* criteria are the best performing individually, but when they are individually

removed, the overall performance is not significantly different. This supports the finding that the signal they bring is very overlapping. On the other side, the TL criterion was the worst performing individually but, when individually removed, it leads to a decrease of the overall accuracy greater than 5%. As regards the linguistic analysis adopted for processing the passages of text returned by the QA module, keyword-based representations have the best performance, followed by stems and lemmas which behave similarly. This was already observed by removing one configuration at a time, where the ablation of configurations using representations based on stems or lemmas did not lead to a decrease of accuracy. As regards the question expansion mechanism, methods adopting it shows a slightly better performance (less than 1%) than those not adopting it, and this is in line with the performance of the single configurations, in which we observed that the question expansion does not seem to have impact on the accuracy of the system. Finally, the configurations using a fewer number of passages (from 1 to 5) are better than those using more passages (from 10 to 30). This seems to contradict the finding stemmed from the analysis of the top-15 configurations (Table 7.2 and Table 7.3), where the best performance were obtained using 15 to

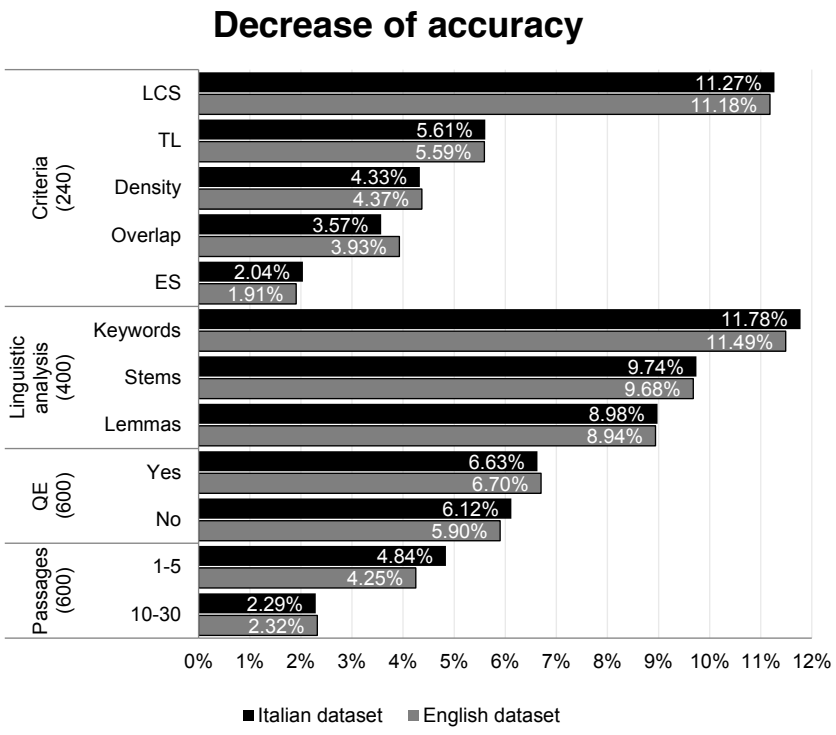


Figure 7.6: Decrease of accuracy for ablation of feature groups.

For each group, the number of features removed is also reported.

30 passages. A possible interpretation is that the information overlap existing using 15 to 30 passages is higher than that existing using 1 to 5 passages, and this led the former combination to be less effective than the latter.

Per Level Analysis of System and Human Performance

We compared the performance of QA and Answer Scoring with that of human players to provide answers to questions at different levels of difficulty.

Playing successfully the WWBM game heavily depends on the player's knowledge about popular culture, hence the comparison with the human players is only performed for the Italian dataset. To this purpose we involved 98 human players, selected using the availability sampling strategy [Singleton and Straits, 1993] (Italian students or graduates). The dataset of 1,960 Italian questions was randomly split into 98 disjoint sets of 20 questions each; each set was assigned to a different user, who provided the answers without having the possibility to consult the Web or other knowledge sources (the level of each question was not disclosed to the users).

As baseline we queried Google with each question and we retrieved the top-30 snippets of text returned by the search engine. Hence, we computed a score for each candidate answer by multiplying the number of times the answer occurred in each snippet with the inverse of the rank of the snippet. Finally, we selected the candidate answer with the highest score (a random selection was adopted to break ties). We refer to this as *Google baseline*.

As Google queries the whole web, while we use only information available in Wikipedia, we decided to compare against a more fair baseline as well. We queried Google restricting the results only to pages coming from Wikipedia, retrieving the top-30 snippets and scoring them in the same way as the *Google baseline*. We refer to this as We refer to this as *Google Wikipedia baseline*.

Figure 7.7 and Figure 7.8 report, for each level of the game:

1. the accuracy of the best configuration obtained by the learning to rank strategy for Italian and English, whose average accuracy is 79.64% (σ = 6.07%) for Italian and 76.41% (σ = 1.45%) for English;

2. the accuracy of the *Google baseline*, which is 67.13% for Italian and 71.80% for English;

3. the accuracy of the *Google Wikipedia baseline*, which is 51.7% for Italian and 60.65% for English;

4. the accuracy achieved by the human players, which is 51.33% (σ = 17.61%) for Italian.

The system's improvement over *Google baseline*, *Google Wikipedia baseline* and human players is statistically significant with $p << 0.0001$, regardless of the language.

The system has quite similar performance for all the levels of the game. As regards the Italian dataset, the best performance is obtained for level 2, while the worst is obtained for levels 7 and 13. However, the error analysis

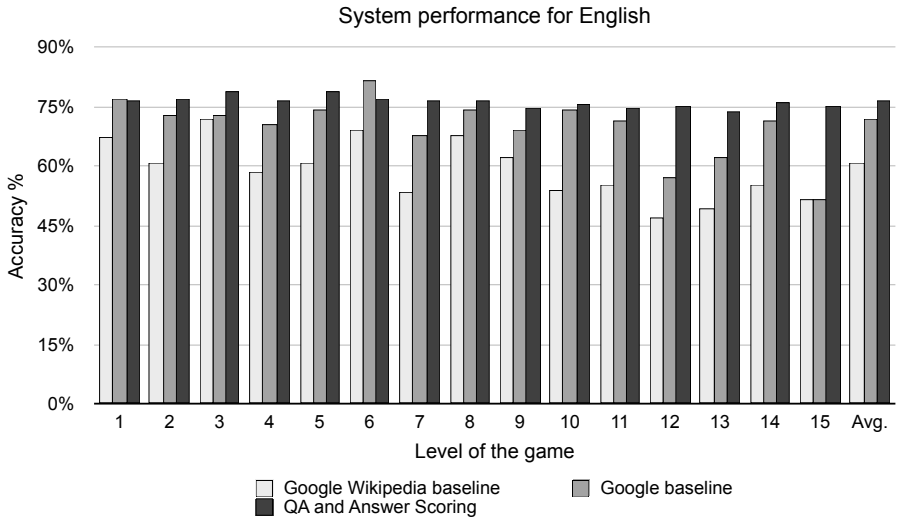


Figure 7.7: Per level accuracy of system and human performance for English.

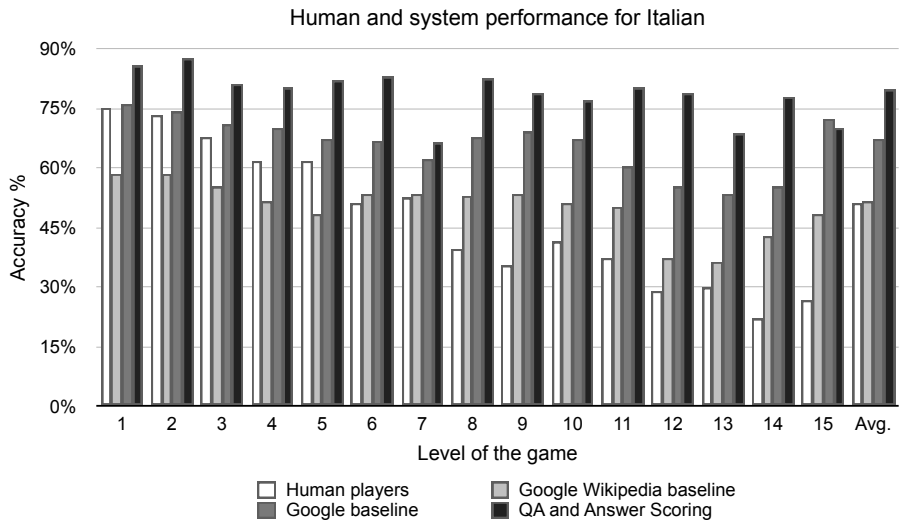


Figure 7.8: Per level accuracy of system and human performance for Italian.

did not report any specific problem for those levels: we only observed a higher concentration of questions the system is not able to deal with (see Section 7.8.1) in one of the five folds. As regards English, the performance is almost constant for all the levels of the game (indeed the standard deviation is very low).

Accuracy of human players (for Italian) decreases almost monotonically, with the best performance obtained on the first level of the game, and the worst on level 14. This observation is coherent with the fact that lower levels of the game correspond to easier questions, while higher levels correspond to more complex questions for which a deeper knowledge is required to provide the correct answers.

The primacy of the system (for Italian) is evident: it significantly outperforms humans for all the 15 levels of the game (not only on average). It is worth to analyze the performance of the players for groups of questions from 1 to 5, 6 to 10, and 11 to 15, respectively, when they reach the guarantee points where the money earned is banked. As regards the first group of questions, humans obtain the worst performance on the (fourth and) fifth question (61.88%), and this is not surprising since this is the first point in which players have the guarantee to keep the earned money. Even though the level of the questions is not disclosed to human players, our hypothesis is that milestone questions are likely more difficult, due to the way the game is designed. Surprisingly, this behavior is not observed for the second milestone question, albeit the performance on that question is below the average. Results obtained by humans on the last group of questions are all below the average, and not comparable with those obtained by the system (absolute difference in performance ranging from +38% to +56%).

The *Google baseline* is always better than human players (for Italian), but it is less accurate than the system (absolute difference in performance amounts to −12.51% for Italian, and −4.61% for English, on average).

The *Google Wikipedia baseline* is less accurate than the *Google baseline* for both Italian and English, and is just slightly more accurate than human players on average (for Italian).

To sum up, the artificial player built using the QA and Answer Scoring modules has the potential to beat human players when playing a real game with its rules, since it is able to correctly answer to questions at different levels of difficulty. We observe very similar performance regardless of the language, i.e. the Question Answering framework and the Answer Scoring criteria work in the same way for Italian and English. The small differences in performance are likely due to the different number of documents extracted from Wikipedia (almost one million for Italian, and three millions for English), which could have impact on the performance of the search engines in the QA framework, and of course to the fact that the two datasets are not comparable.

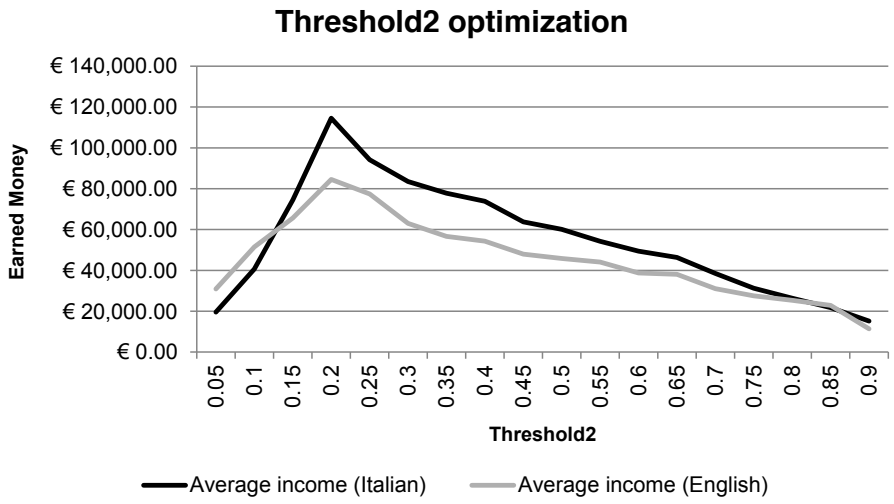


Figure 7.9: Plot of the average income for different values of the threshold.

However, playing a real game is a very complex task, since it requires a proper strategy to manage the lifelines, to decide whether to answer to a question or to retire from the game by taking the earned money. This is the purpose of the experiment described in the next section.

7.8.2 Experiment 2: Evaluation of the Artificial Player

The goal of this experiment is to evaluate the ability of the artificial player to play the game, by implementing a proper strategy to use the lifelines, to decide whether to answer to a question even in a condition of uncertainty or to retire from the game by taking the earning money. We ran the experiment by comparing the performance of the human players with the performance achieved by the best system configuration obtained by using the learning to rank approach, and whose strategy to play the game is defined by the decision making algorithm described in Section 7.7. The comparison is carried out by evaluating the level reached during the game and the money earned by the players[5]. As in the previous experiment, the comparison between the performance of the artificial player and humans is carried out only for Italian.

We involved 35 subjects, different from those involved in the previous experiment, who overall played 325 games. Each game consists of questions selected by the Game Manager, which randomly sampled one question for each level as the game proceeds (games played by the same human player always have

[5]The Italian and English versions of the game have a different currency and a different distribution of monetary values for each level of the game. For the sake of comparison we decided to use the same currency and monetary values as for Italian.

different questions for the same level). On the other side, the artificial player played 160 games, whose questions are also selected by the Game Manager from both the Italian and English datasets. An important setting to configure the artificial player is the value of the two thresholds used by the decision making algorithm for assessing a situation of uncertainty (Section 7.7). $Threshold_1$ is used to evaluate the score of the best candidate answer, while $threshold_2$ is used to evaluate the difference between the score of the best and the second best candidate answer. We decided to assign 0 to $threshold_1$, i.e. answers with a score different from zero are always trusted and used by the artificial player. The value of $threshold_2$ was optimized by empirically varying its value, and selecting the one which led the artificial player to obtain the highest average income on the questions in the validation set of the first fold. $Threshold_2$ was finally set to 0.2 for both Italian and English (plot showing how different values affect the average income depicted in Figure 7.9).

Figure 7.10 shows the boxplot of the levels reached during the game and Figure 7.11 the boxplot of the money earned by the players, while Figures 7.12 and 7.13 report the distribution of games reaching a specific level and the distribution of games ended with the income in a specific interval.

All the players are able to reach the last level of the game, but the average level reached by humans is between five and six (5.65), with respect to the artificial player which reaches a level between seven and eight, i.e. 7.88 and 7.60 for Italian and English, respectively. This is also highlighted by the median value, which is the fifth level for humans and the seventh for the artificial player.

More than half of the times (51.38%) the human players ended the game by reaching levels from 1 to 5, while 40% of times reached levels 6 to 10. Few times (8.62%) humans were able to reach the last levels (level 15 reached only once). The distribution of the games ended by the artificial player in the three groups of questions is almost uniform for both Italian and English (with a slightly better performance for Italian), and this is coherent with the results in Figure 7.7 and Figure 7.8, in which the accuracy of the system is very similar for all the levels of the game.

301 out of 325 games (92.61%) played by humans ended due to an error in the response, while 24 times (7.39%) the players ended the game by retiring and taking the earned money (in three cases the players retired from the game at level 6 and 11, even though a wrong answer would not have any effect on the earned money). Moreover, human players never ended the game with the maximun prize. Differently from humans, the artificial player was able to successfully complete the game. Indeed, it earned €1,000,000 17 times (10.62%) for Italian, and 12 times (7.50%) for English. 116 games (72.50%) ended due to a wrong answer by the artificial player for Italian, while 27 times (16.87%) it retired from the game without providing the answer. The artificial player for English

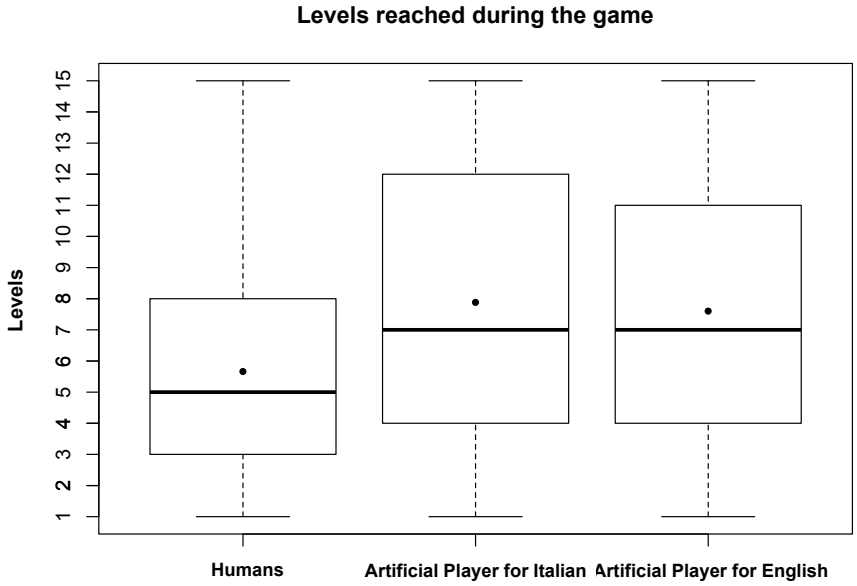


Figure 7.10: Distribution of the levels reached during the game by the players.

Upper and lower ends of the boxes represent the 3^{rd} and 1^{st} quartile, respectively. Whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range. Median values are depicted with solid lines, mean values with solid points.

ended 114 games (71.25%) due to a wrong answer, and retired from the game 34 times (21.25%). The highest percentage of games ended by the artificial player without providing the answer highlights its more conservative and low risk strategy. It is interesting to note that the decision making algorithm does not allow the artificial player to end the game at level 6 or 11.

The money earned by humans is €5,926 on average, while the average income of the artificial player is significantly higher. Indeed, it earned €114,531 for Italian, and €88,878 for English. The detailed figures are shown in Figure 7.13, where the games ended with a zero income are reported, as well as the games ended in each interval corresponding to milestone questions. Most of the games played by humans (88.30%) ended with a zero income or by answering questions in the first group. This means that they reached the last two groups of questions 11.70% of times, differently from the artificial player which reached the last two groups of questions about 40% of times for both Italian and English. The fewer percentage of games ended with a zero income by the artificial player confirms its risk averse behavior (albeit some differences between Italian and English exist) that, coupled with its ability to provide correct answers regardless of the

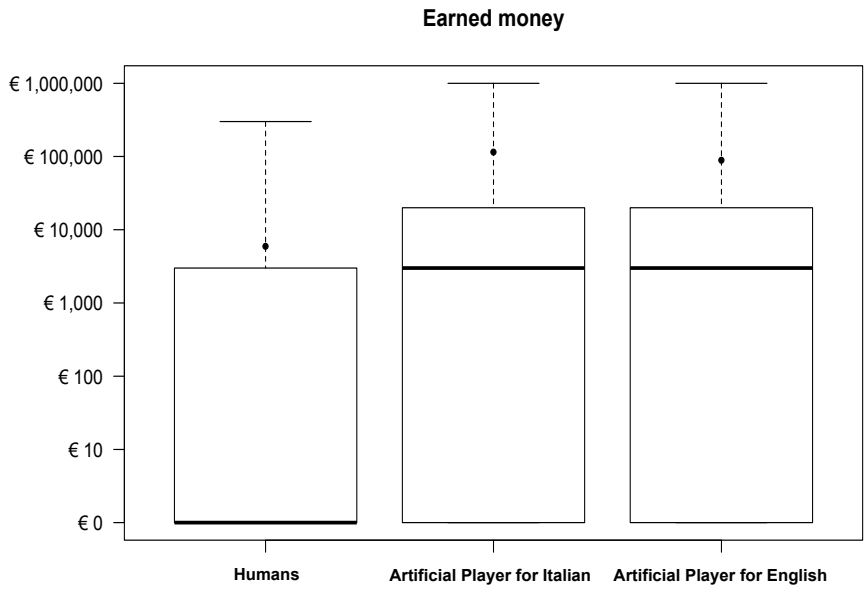
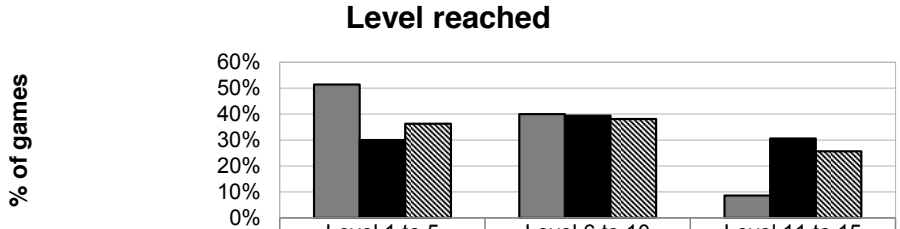


Figure 7.11: Distribution of the money earned by the players (in log scale).

Upper and lower ends of the boxes represent the 3^{rd} and 1^{st} quartile, respectively. Whiskers extend to the most extreme data point which is no more than 1.5 times the interquartile range. Median values are depicted with solid lines, mean values with solid points.



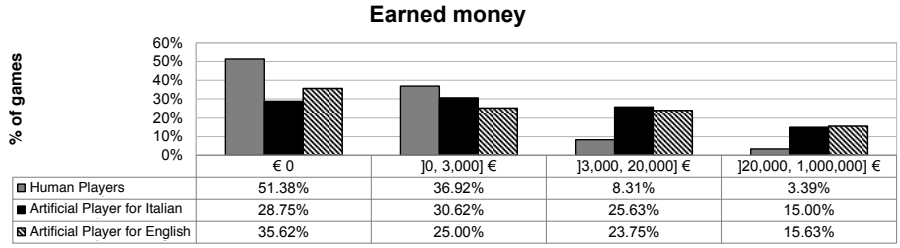
	Level 1 to 5	Level 6 to 10	Level 11 to 15
Human Players	51.38%	40.00%	8.62%
Artificial Player for Italian	30.00%	39.38%	30.62%
Artificial Player for English	36.25%	38.13%	25.62%

Figure 7.12: Distribution of games reaching a specific level.

level of the game, allows it to end the games with a higher average income.

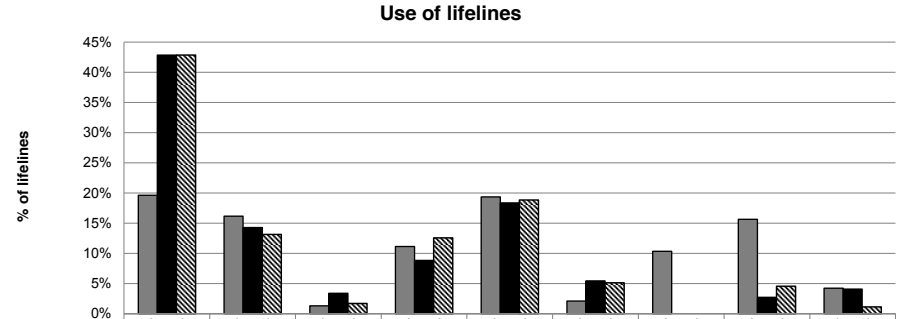
Figure 7.14 reports the use of the lifelines during the different stages of the game.

The comparison of the strategies adopted by humans and the artificial player



	€ 0	]0, 3,000] €	]3,000, 20,000] €	]20,000, 1,000,000] €
Human Players	51.38%	36.92%	8.31%	3.39%
Artificial Player for Italian	28.75%	30.62%	25.63%	15.00%
Artificial Player for English	35.62%	25.00%	23.75%	15.63%

Figure 7.13: Distribution of games ended with the income in a specific interval.



	Level 1 to 5	Level 6 to 10	Level 11 to 15	Level 1 to 5	Level 6 to 10	Level 11 to 15	Level 1 to 5	Level 6 to 10	Level 11 to 15
	Poll the Audience			Phone a Friend			50-50		
Humans	19.63%	16.18%	1.33%	11.14%	19.36%	2.12%	10.34%	15.65%	4.24%
AP for Italian	42.86%	14.29%	3.40%	8.84%	18.37%	5.44%	0.00%	2.72%	4.08%
AP for English	42.86%	13.14%	1.71%	12.57%	18.86%	5.14%	0.00%	4.57%	1.14%

Figure 7.14: Distribution of the lifelines used during the game.

is not fair since the latter uses a static order for the available lifelines, as defined by the decision making algorithm (i.e. *Poll the Audience, Phone a Friend, 50:50*). Despite this limitation, overall, the support provided by the lifelines to the artificial player is valuable. Indeed, the artificial player is able to provide the correct answer 114 times for Italian and 141 times for English thanks to the lifelines. Human players and the artificial player use a single lifeline per game on average, hence they behave in a very similar way. During the first stages of the game, the frequency of usage of *Poll the Audience* by the artificial player (42.86% both for Italian and for English) is as much as the cumulative frequency of use of all the three available lifelines by the human players (41.11%). This means that the artificial player is forced to use that lifeline before the others, as defined in the decision making algorithm, while the human players use all the available lifelines, even though they prefer *Poll the Audience* during the first stages of the game. The frequency of use of *Phone a Friend* by humans and the artificial player is quite similar. As expected, *50:50* is never used by the artificial player during the first levels of the game, while humans uniformly used

it in all the levels of the game.

To sum up, as expected from the results of Experiment 1, the artificial player is able to outperform humans, even though it adopts a very simple decision making strategy. The better performance is in terms of both the average reached level and the earned money at the end of the game. The performance of the artificial player for Italian and English is pretty much the same, except the average income, which is significantly better for Italian than English. As observed for the performance of the QA and Answer Scoring, this is likely due to the different size of the knowledge sources for Italian and for English, as well as of course to the different datasets used in the experiment.

7.9 Summary

We proposed the architecture of an artificial player for the language game "Who Wants to Be a Millionaire?", based on the following modules:

- Question Answering: it is able to retrieve passages of text relevant to a specific question expressed in natural language, by using Wikipedia and DBpedia open knowledge sources;
- Answer Scoring: it implements several heuristics based on the analysis of the results returned by the Question Answering module, in order to assign a score to the four candidate answers;
- Decision Making: it chooses the strategy to play the game, by exploiting the scores of the four candidate answers, the availability of lifelines, and the current level of the game.

Hence, we can provide an answer to both Research Question 3 (RQ3) and Research Question 4 (RQ4). As regards RQ3, this work actually led to the definition of an effective language-independent framework for QA and answer validation able to leverage Wikipedia and DBpedia open knowledge sources, as well as to an effective strategy to combine different criteria for scoring candidate answers through machine learning techniques.

As regards RQ4, using the Question Answering and Answer Scoring modules in RQ1 we were able to build an artificial player which outperforms human players in terms of average accuracy in correctly answering to questions of the WWBM game, and in terms of ability to play real games with their rules.

Part IV

Conclusion

8

(Research) Question Answering

> *"Forty-two!" yelled Loonquawl. "Is that all you've got to show for seven and a half million years' work?"*
> *"I checked it very thoroughly," said the computer, "and that quite definitely is the answer. I think the problem, to be quite honest with you, is that you've never actually known what the question is."*
>
> – Douglas Adams, *The Hitchhiker's Guide to the Galaxy*

Throughout the chapters of this thesis we tried to answer the four research questions in Section 1.1.

In particular, in Chapter 5 and Chapter 6 we contribute to bring order to the vast literature on the task of best answer selection by gathering a big number of features, grouped in five families, combining them with a learning to rank approach, and testing them on large datasets from *Yahoo! Answers*. We propose a new suite of Distributional-Semantics-based features, in combination with the textual ones and the information from several expertise networks. Besides being able to outperform the prediction ability of state-of-the-art methods up to 27% in P@1, our experiments allow us also to draw important conclusions about the impact of different features employed that have never been spell out in previous literature due to a lack of extensive and systematic feature comparison. We summarize our findings as follows.

- *Textual features are by far the ones with higher predictive potential*, compared to user-centric features or to the expertise network centrality scores. This is mainly due to the fact that the content of the question and answers (their topic and structure) are a more important source of information to determine the question-answering match rather than the expertise of the

answerers. Those features are to prefer when dealing with factual-type questions.

- Among the textual features, *text quality and Distributional Semantics are in general to prefer to linguistic similarity.* We indeed found that linguistic similarity's signal is mostly captured by other features already. This is an important finding as linguistic similarity features have been used in a number of previous approaches but are roughly 12 times more computationally expensive than Distributional Semantics based ones. This answers RQ2.

- *The new Distributional Semantics based features proposed achieve surprisingly good results* considering their small number. This answers RQ1.

- User and network features determine a considerable improvement over the textual-based features and their contribution is not completely overlapping, meaning that *considering network interaction rather than the individual user activity adds real value to the prediction.*

We believe our work will help to take the stock of the research on the task of best answer prediction and set the basis for new developments in the field.

In Chapter 7, we proposed the architecture of an artificial player for the language game "Who Wants to Be a Millionaire?", based on the following modules:

- Question Answering: is able to retrieve passages of text relevant to a specific question expressed in natural language, by using Wikipedia and DBpedia open knowledge sources;

- Answer Scoring: implements several heuristics based on the analysis of the results returned by the Question Answering module, in order to assign a score to the four candidate answers;

- Decision Making: chooses the strategy to play the game, by exploiting the scores of the four candidate answers, the availability of lifelines, and the current level of the game.

Hence, we can provide an answer to both RQ3 and RQ4. As regards RQ3, this work actually led to the definition of an effective language-independent framework for QA and answer validation able to leverage Wikipedia and DBpedia open knowledge sources, as well as to an effective strategy to combine different criteria for scoring candidate answers through machine learning techniques.

As regards RQ4, using the Question Answering and Answer Scoring modules in RQ1 we were able to build an artificial player which outperforms human players in terms of average accuracy in correctly answering to questions of the WWBM game, and in terms of ability to play real games with their rules.

8.1 Future Work

The experiments described in this thesis helped to answer the research questions we were aiming to answer. At the same time there are sill several directions that need to be investigated and a lot of future work to do.

From an experimental point of view, there are two major points to address for getting an even more detailed insight about the usefulness of DSMs in QA.

The first is to analyze the impact of the model construction methods on the overall performance. Building models adopting bigrams and trigrams and not only unigrams could affect the ability to incorporate named entities inside the semantic space. Experimenting with different numbers of dimensions, maybe as a function of the corpus size, is another interesting analysis to carry out, as is comparing models built on simple terms with models built adopting different lexicalization levels, like stems, lemmas or even word senses.

The second relevant point is the semantic compositionality. Our proposed approach was to sum the vectors of terms present in questions and answers in order to have a composite representation, but several other operators can be used for this task. We stuck with the sum because no clear winner emerged in literature as the best way to compose term vectors, and the datasets adopted to evaluate them reported really low agreement among the human assessors. Because of this we decided to follow the simplest and more efficient option and obtained good results. Nevertheless, a clear and wide-scope study on semantic compositionality would be really beneficial for both scientific understanding and practical implications, like improved performance in our experiments.

A way to integrate factoid and non-factoid question answering techniques in a coherent approach in order to cover a really wide range of possible questions would be a really compelling future research. Our proposal for merging QA over linked open data with QA over unstructured text is a step in that direction, but several other steps should be done in order to full address all remaining issues. For example, speaking more in general of the weaknesses of our proposals, our approach does not consider any form of reasoning, while a really open-domain QA system would probably need some kind of reasoning (maybe abductive) in order to answer specific kinds of complex questions, where the system has no complete knowledge and tries to guess an answer. In our approach the QA system knows only what is available in the corpus used for finding answers, while using this knowledge to guess novel answers, maybe merging available answers, maybe doing inference on them, would for sure be an interesting future work to address.

Another fascinating and thought-provoking direction that we foresee has roots in the fact that QA systems have no preconceptions on the answers they give, they just use evidence in the form of text and data, and this leads the users of these systems to possible serendipitous discoveries. Deepen this aspect

of QA systems is interesting from both a humanist and psychological point of view and not only from a technical point of view.

Finally, we think that a multidisciplinary approach to the analysis of the possible implications of the actual and future QA technology is of absolute importance. Like every technology that radically changes how humans actually do things, QA technology is seen as exciting for its time-saving knowledge-enriching business-enabling decision support potential, but it is also seen as frightening for the possible abuses like replacing human judgment, taking away jobs and global surveillance related fears. This opens up many questions to be answered from a philosophical, economical, legal, sociological, psychological and ethical point of view.

Bibliography

David A. Ferrucci, Anthony Levas, Sugato Bagchi, David Gondek, and Erik T. Mueller. Watson: Beyond jeopardy! *Artificial Intelligence*, 199:93–105, 2013.

David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010.

Ludwig Wittgenstein. *Philosophische Untersuchungen*. Suhrkamp Verlag, Frankfurt am Main, 1953.

Piero Molino and Pierpaolo Basile. Questioncube: a framework for question answering. In Giambattista Amati, Claudio Carpineto, and Giovanni Semeraro, editors, *Proceedings of the 3rd Italian Information Retrieval Workshop, Bari, Italy, January 26-27, 2012*, volume 835 of *CEUR Workshop Proceedings*, pages 167–178. CEUR-WS.org, 2012.

Piero Molino, Pierpaolo Basile, Annalina Caputo, Pasquale Lops, and Giovanni Semeraro. Exploiting distributional semantic models in question answering. In *Sixth IEEE International Conference on Semantic Computing, ICSC 2012, Palermo, Italy, September 19-21, 2012*, pages 146–153. IEEE Computer Society, 2012. ISBN 978-1-4673-4433-3.

Piero Molino and Luca Maria Aiello. Distributed representations for semantic matching in non-factoid question answering. In Julio Gonzalo, Hang Li, Alessandro Moschitti, and Jun Xu, editors, *Proceedings of Workshop on Semantic Matching in Information Retrieval co-located with the 37th international ACM SIGIR conference on research and development in information retrieval, SMIRSIGIR 2014, Queensland, Australia, July 11, 2014.*, volume 1204 of *CEUR Workshop Proceedings*, pages 38–45. CEUR-WS.org, 2014.

Piero Molino, Pierpaolo Basile, Ciro Santoro, Pasquale Lops, Marco de Gemmis, and Giovanni Semeraro. A virtual player for "who wants to be a millionaire?" based on question answering. In Matteo Baldoni, Cristina Baroglio, Guido Boella, and Roberto Micalizio, editors, *AI*IA 2013: Advances in Artificial Intelligence - XIIIth International Conference of the Italian Association for Artificial Intelligence, Turin, Italy, December 4-6, 2013. Proceedings*, volume 8249 of *Lecture Notes in Computer Science*, pages 205–216. Springer, 2013a. ISBN 978-3-319-03523-9.

Piero Molino, Pierpaolo Basile, Annalina Caputo, Pasquale Lops, and Giovanni Semeraro. Distributional semantics for answer re-ranking in question answering. In Roberto Basili, Fabrizio Sebastiani, and Giovanni Semeraro, editors, *Proceedings of the 4th Italian Information Retrieval Workshop, Pisa, Italy, January 16-17, 2013*, volume 964 of *CEUR Workshop Proceedings*, pages 100–103. CEUR-WS.org, 2013b.

Piero Molino. Semantic models for answer re-ranking in question answering. In Jones et al. [2013], page 1146. ISBN 978-1-4503-2034-4.

Ellen M. Voorhees and Dawn M. Tice. The TREC-8 question answering track evaluation. In *TREC*, 1999.

Anselmo Peñas, Pamela Forner, Álvaro Rodrigo, Richard F. E. Sutcliffe, Corina Forascu, and Cristina Mota. Overview of respubliqa 2010: Question answering evaluation over european legislation. In Braschler et al. [2010]. ISBN 978-88-904810-0-0.

Bert F. Jr. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. Baseball, an automatic question-answerer. *Managing Requirements Knowledge, International Workshop on*, 0:219, 1961.

Robert F. Simmons. Answering english questions by computer: a survey. *Communumintions of the ACM*, 8(1):53–70, 1965.

David A. Ferrucci. Ibm's watson/deepqa. *SIGARCH Computer Architecture News*, 39(3), 2011.

Susan T. Dumais, Michele Banko, Eric Brill, Jimmy J. Lin, and Andrew Y. Ng. Web question answering: is more always better? In *SIGIR 2002: Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, August 11-15, 2002, Tampere, Finland*, pages 291–298. ACM, 2002.

Jimmy J. Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Transactions on Information Systems*, 25(2), 2007.

Sanda M. Harabagiu, Marius Paşca, and Steven J. Maiorano. Experiments with open-domain textual question answering. In *COLING 2000, 18th International Conference on Computational Linguistics, Proceedings of the Conference, 2 Volumes, July 31 - August 4, 2000, Universität des Saarlandes, Saarbrücken, Germany*, pages 292–298. Morgan Kaufmann, 2000a.

Marius Paşca and Kiril Ribarov. *Review: Open-Domain Question Answering from Large Text Collections*, volume 81. Kluwer Academic Publishers, Hingham, MA, USA, 2004.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. Using syntactic information for improving why-question answering. In Donia Scott and Hans Uszkoreit, editors, *COLING 2008, 22nd International Conference on Computational Linguistics, Proceedings of the Conference, 18-22 August 2008, Manchester, UK*, pages 953–960, 2008. ISBN 978-1-905593-44-6.

Sanda M. Harabagiu, Dan I. Moldovan, Marius Paşca, Rada Mihalcea, Mihai Surdeanu, Razvan C. Bunescu, Roxana Girju, Vasile Rus, and Paul Morarescu. FALCON: boosting knowledge for answer engines. In *TREC*, 2000b.

Eduard H. Hovy, Laurie Gerber, Ulf Hermjakob, Michael Junk, and Chin-Yew Lin. Question answering in webclopedia. In *TREC*, 2000.

Jiangping Chen, Anne Diekema, Mary D. Taffet, Nancy J. McCracken, Necati Ercan Ozgencil, Ozgur Yilmazel, and Elizabeth D. Liddy. Question answering: CNLP at the TREC-10 question answering track. In *TREC*, 2001.

Dan I. Moldovan, Marius Pasca, Sanda M. Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems*, 21(2):133–154, 2003.

Xin Li and Dan Roth. Learning question classifiers: the role of semantic information. *Natural Language Engineering*, 12(3):229–249, 2006.

Suzan Verberne, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. What is not in the bag of words for *Why*-qa? *Computational Linguistics*, 36(2):229–245, 2010.

Ryuichiro Higashinaka and Hideki Isozaki. Corpus-based question answering for why-questions. In *Third International Joint Conference on Natural Language Processing, IJCNLP 2008, Hyderabad, India, January 7-12, 2008*, pages 418–425. The Association for Computer Linguistics, 2008.

Radu Soricut and Eric Brill. Automatic question answering using the web: Beyond the factoid. *Information Retrieval*, 9(2):191–206, 2006.

Eugene Agichtein, Carlos Castillo, Debora Donato, Aristides Gionis, and Gilad Mishne. Finding high-quality content in social media. In Najork et al. [2008], pages 183–194.

Mihai Surdeanu, Massimiliano Ciaramita, and Hugo Zaragoza. Learning to rank answers to non-factoid questions from web collections. *Computational Linguistics*, 37(2):351–383, 2011.

Arvind Agarwal, Hema Raghavan, Karthik Subbian, Prem Melville, Richard D. Lawrence, David Gondek, and James Fan. Learning to rank for robust question answering. In Xue-wen Chen, Guy Lebanon, Haixun Wang, and Mohammed J. Zaki, editors, *21st ACM International Conference on Information and Knowledge Management, CIKM'12, Maui, HI, USA, October 29 - November 02, 2012*, pages 833–842. ACM, 2012. ISBN 978-1-4503-1156-4.

Tie-Yan Liu. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval*, 3(3):225–331, 2009.

Suzan Verberne, Hans van Halteren, Daphne Theijssen, Stephan Raaijmakers, and Lou Boves. Learning to rank for *why*-question answering. *Information Retrieval*, 14(2):107–132, 2011.

Vasin Punyakanok, Dan Roth, Wen-tau Yih, and Dav Zimak. Semantic role labeling via integer linear programming inference. In *COLING 2004, 20th International Conference on Computational Linguistics, Proceedings of the Conference, 23-27 August 2004, Geneva, Switzerland*, 2004.

Libin Shen and Aravind K. Joshi. Ranking and reranking with perceptron. *Machine Learning*, 60(1-3):73–96, 2005.

Renxu Sun, Jing Jiang, Yee Fan Tan, Hang Cui, Tat-Seng Chua, and Min-Yen Kan. Using syntactic and semantic relation analysis in question answering. In Voorhees and Buckland [2005].

Nico Schlaefer, Jeongwoo Ko, Justin Betteridge, Manas A. Pathak, Eric Nyberg, and Guido Sautter. Semantic extensions of the ephyra QA system for TREC 2007. In Ellen M. Voorhees and Lori P. Buckland, editors, *Proceedings of The Sixteenth Text REtrieval Conference, TREC 2007, Gaithersburg, Maryland, USA, November 5-9, 2007*, volume Special Publication 500-274. National Institute of Standards and Technology (NIST), 2007.

Matthew W. Bilotti. Linguistic and semantic passage retrieval strategies for question answering. *SIGIR Forum*, 44 (2):83, 2010.

Lorand Dali, Delia Rusu, Blaz Fortuna, Dunja Mladenic, and Marko Grobelnik. Question answering based on semantic graphs. In *Proceedings of the Workshop on Semantic Search (Sem-Search 2009)*, 2009.

Aliaksei Severyn and Alessandro Moschitti. Structural relationships for large-scale learning of answer re-ranking. In Hersh et al. [2012], pages 741–750. ISBN 978-1-4503-1472-5.

Abdessamad Echihabi and Daniel Marcu. A noisy-channel approach to question answering. In Hinrichs and Roth [2003], pages 16–23.

Peter F. Brown, Stephen Della Pietra, Vincent J. Della Pietra, and Robert L. Mercer. The mathematics of statistical machine translation: Parameter estimation. *Computational Linguistics*, 19(2):263–311, 1993.

Hang Cui, Renxu Sun, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Question answering passage retrieval using dependency relations. In Ricardo A. Baeza-Yates, Nivio Ziviani, Gary Marchionini, Alistair Moffat, and John Tait, editors, *SIGIR 2005: Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Salvador, Brazil, August 15-19, 2005*, pages 400–407. ACM, 2005. ISBN 1-59593-034-5.

Eugene Agichtein, Steve Lawrence, and Luis Gravano. Learning search engine specific query transformations for question answering. In Vincent Y. Shen, Nobuo Saito, Michael R. Lyu, and Mary Ellen Zurko, editors, *Proceedings of the Tenth International World Wide Web Conference, WWW 10, Hong Kong, China, May 1-5, 2001*, pages 169–178. ACM, 2001. ISBN 1-58113-348-0.

Vanessa Murdock and W. Bruce Croft. A translation model for sentence retrieval. In *HLT/EMNLP 2005, Human Language Technology Conference and Conference on Empirical Methods in Natural Language Processing, Proceedings of the Conference, 6-8 October 2005, Vancouver, British Columbia, Canada*. The Association for Computational Linguistics, 2005.

Xiaobing Xue, Jiwoon Jeon, and W. Bruce Croft. Retrieval models for question and answer archives. In Myaeng et al. [2008], pages 475–482. ISBN 978-1-60558-164-4.

Stefan Riezler, Alexander Vasserman, Ioannis Tsochantaridis, Vibhu O. Mittal, and Yi Liu. Statistical machine translation for query expansion in answer retrieval. In John A. Carroll, Antal van den Bosch, and Annie Zaenen, editors, *ACL 2007, Proceedings of the 45th Annual Meeting of the Association for Computational Linguistics, June 23-30, 2007, Prague, Czech Republic*. The Association for Computational Linguistics, 2007.

Nico Schlaefer, P. Gieselman, and Guido Sautter. The ephyra QA system at TREC 2006. In Voorhees and Buckland [2006].

Christiane D. Fellbaum. *WordNet: an electronic lexical database*. Language, speech, and communication. MIT Press, 1998. ISBN 9780262061971.

Eros Zanchetta and Marco Baroni. Morph-it! a free corpus-based morphological resource for the italian language. *Corpus Linguistics 2005*, 1(1), 2005. ISSN 1747-9398.

Corinna Cortes and Vladimir Vapnik. Support-vector networks. *Machine Learning*, 20(3):273–297, 1995.

Taku Kudo and Yuji Matsumoto. Fast methods for kernel-based text analysis. In Hinrichs and Roth [2003], pages 24–31.

Eneko Agirre and Aitor Soroa. Personalizing pagerank for word sense disambiguation. In Alex Lascarides, Claire Gardent, and Joakim Nivre, editors, *EACL 2009, 12th Conference of the European Chapter of the Association for Computational Linguistics, Proceedings of the Conference, Athens, Greece, March 30 - April 3, 2009*, pages 33–41. The Association for Computer Linguistics, 2009.

Sergey Brin and Lawrence Page. The anatomy of a large-scale hypertextual web search engine. *Computer Networks*, 30(1-7):107–117, 1998.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. The Stanford CoreNLP natural language processing toolkit. In *Proceedings of 52nd Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 55–60, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

Jinho D. Choi. *Optimization of Natural Language Processing Components for Robustness and Scalability*. PhD thesis, Computer Science and Cognitive Science, Boulder, CO, USA, 2012. AAI3549172.

Massimiliano Ciaramita and Yasemin Altun. Broad-coverage sense disambiguation and information extraction with a supersense sequence tagger. In Dan Jurafsky and Éric Gaussier, editors, *EMNLP 2007, Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, 22-23 July 2006, Sydney, Australia*, pages 594–602. ACL, 2006. ISBN 1-932432-73-6.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Gerard M. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Communnications of ACM*, 18:613–620, November 1975. ISSN 0001-0782.

ChengXiang Zhai and John D. Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 334–342. ACM, 2001. ISBN 1-58113-331-6.

Claudio Carpineto, Renato de Mori, Giovanni Romano, and Brigitte Bigi. An information-theoretic approach to automatic query expansion. *ACM Transactions on Information Systems*, 19(1):1–27, 2001.

Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In Fabio Crestani, Stéphane Marchand-Maillet, Hsin-Hsi Chen, Efthimis N. Efthimiadis, and Jacques Savoy, editors, *Proceeding of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2010, Geneva, Switzerland, July 19-23, 2010*, pages 579–586. ACM, 2010. ISBN 978-1-4503-0153-4.

Gianni Amati and C. J. van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Transactions on Information Systems*, 20(4):357–389, 2002.

Christof Monz. Minimal Span Weighting Retrieval for Question Answering. In Rob Gaizauskas, Mark Greenwood, and Mark Hepple, editors, *Proceedings of the SIGIR 2004 Workshop on Information Retrieval for Question Answering*, pages 23–30, 2004.

Edward A. Fox and Joseph A. Shaw. Combination of multiple searches. In *TREC*, pages 243–252, 1993.

Thomas K. Landauer and Susan T. Dumais. A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge. *Psychological Review*, 104:211–240, 1997.

Curt Burgess, Kay Livesay, and Kevin Lund. Explorations in context space: Words, sentences, discourse. *Discourse Processes*, 25(2-3):211–257, 1998.

Michael N. Jones and Douglas J. K. Mewhort. Representing word meaning and order information in a composite holographic lexicon. *Psychological Review*, 114(1):1–37, 2007.

Peter D. Turney. Similarity of semantic relations. *Computational Linguistics*, 32(3):379–416, 2006.

Peter D. Turney and Michael L. Littman. Corpus-based learning of analogies and semantic relations. *Machine Learning*, 60(1-3):251–278, 2005.

Michael B.W. Wolfe, M.E. Schreiner, Bob Rehder, Darrell Laham, Peter W. Foltz, Walter Kintsch, and Thomas K. Landauer. Learning from text: Matching readers and texts by latent semantic analysis. *Discourse Processes*, 25 (2-3):309–336, 1998.

Peter W. Foltz, Darrell Laham, and Thomas K. Landauer. The intelligent essay assessor: Applications to educational technology. *Interactive Multimedia Electronic Journal of Computer-Enhanced Learning*, 1(2), 1999.

Hinrich Schütze and Jan O. Pedersen. Information retrieval based on word senses. In *Proceedings of the 4th Annual Symposium on Document Analysis and Information Retrieval*, pages 161–175, 1995.

Hinrich Schütze. Automatic word sense discrimination. *Computational Linguistics*, 24(1):97–123, 1998.

Pierpaolo Basile. Super-sense tagging using support vector machines and distributional features. In Bernardo Magnini, Francesco Cutugno, Mauro Falcone, and Emanuele Pianta, editors, *Evaluation of Natural Language and Speech Tools for Italian, International Workshop, EVALITA 2011, Rome, Italy, January 24-25, 2012, Revised Selected Papers*, volume 7689 of *Lecture Notes in Computer Science*, pages 176–185. Springer, 2011. ISBN 978-3-642-35827-2.

Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel P. Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12:2493–2537, 2011.

Joseph P. Turian, Lev-Arie Ratinov, and Yoshua Bengio. Word representations: A simple and general method for semi-supervised learning. In Jan Hajic, Sandra Carberry, and Stephen Clark, editors, *ACL 2010, Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics, July 11-16, 2010, Uppsala, Sweden*, pages 384–394. The Association for Computer Linguistics, 2010. ISBN 978-1-932432-66-4.

Peter D. Turney and Patrick Pantel. From frequency to meaning: Vector space models of semantics. *Journal of Artificial Intelligence Research (JAIR)*, 37:141–188, 2010.

Jeff Mitchell and Mirella Lapata. Composition in distributional models of semantics. *Cognitive Science*, 34(8):1388–1429, 2010.

Dominic Widdows and Kathleen Ferraro. Semantic vectors: a scalable open source package and online technology management application. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco* con [2008].

Scott C. Deerwester, Susan T. Dumais, Thomas K. Landauer, George W. Furnas, and Richard A. Harshman. Indexing by latent semantic analysis. *JASIS*, 41(6):391–407, 1990.

Pierpaolo Basile, Annalina Caputo, and Giovanni Semeraro. Integrating sense discrimination in a semantic information retrieval system. In Alessandro Soro, Eloisa Vargiu, Giuliano Armano, and Gavino Paddeu, editors, *Information Retrieval and Mining in Distributed Environments*, volume 324 of *Studies in Computational Intelligence*, pages 249–265. Springer Berlin / Heidelberg, 2011.

Pentti Kanerva. *Sparse Distributed Memory*. MIT Press, 1988.

William B. Johnson and Joram Lindenstrauss. Extensions of lipschitz mappings into a hilbert space. *Conference on Modern Analysis and Probability, Contemporary Mathematics*, 26:189–206, 1984.

S. Dasgupta and A. Gupta. An elementary proof of the johnson-lindenstrauss lemma. Technical report, Technical Report TR-99-006, International Computer Science Institute, Berkeley, California, USA, 1999.

Linus Sellberg and Arne Jönsson. Using random indexing to improve singular value decomposition for latent semantic analysis. In *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco* con [2008].

Tomas Mikolov, Wen-tau Yih, and Geoffrey Zweig. Linguistic regularities in continuous space word representations. In Lucy Vanderwende, Hal Daumé III, and Katrin Kirchhoff, editors, *Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, Proceedings, June 9-14, 2013, Westin Peachtree Plaza Hotel, Atlanta, Georgia, USA*, pages 746–751. The Association for Computational Linguistics, 2013a.

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

Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Janvin. A neural probabilistic language model. *Journal of Machine Learning Research*, 3:1137–1155, 2003.

Paul Smolensky. Tensor product variable binding and the representation of symbolic structures in connectionist systems. *Artificial Intelligence*, 46(1-2):159–216, 1990.

Hang Li. *Learning to Rank for Information Retrieval and Natural Language Processing.* Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers, 2011.

Tie-Yan Liu. *Learning to Rank for Information Retrieval.* Springer, 2011. ISBN 978-3-642-14266-6.

Yoav Freund, Raj D. Iyer, Robert E. Schapire, and Yoram Singer. An efficient boosting algorithm for combining preferences. *Journal of Machine Learning Research*, 4:933–969, 2003.

Christopher J. C. Burges, Tal Shaked, Erin Renshaw, Ari Lazier, Matt Deeds, Nicole Hamilton, and Gregory N. Hullender. Learning to rank using gradient descent. In Luc De Raedt and Stefan Wrobel, editors, *Machine Learning, Proceedings of the Twenty-Second International Conference (ICML 2005), Bonn, Germany, August 7-11, 2005*, volume 119 of *ACM International Conference Proceeding Series*, pages 89–96. ACM, 2005. ISBN 1-59593-180-5.

Yunbo Cao, Jun Xu, Tie-Yan Liu, Hang Li, Yalou Huang, and Hsiao-Wuen Hon. Adapting ranking SVM to document retrieval. In Efthimiadis et al. [2006], pages 186–193. ISBN 1-59593-369-7.

Jun Xu and Hang Li. Adarank: a boosting algorithm for information retrieval. In Wessel Kraaij, Arjen P. de Vries, Charles L. A. Clarke, Norbert Fuhr, and Noriko Kando, editors, *SIGIR 2007: Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Amsterdam, The Netherlands, July 23-27, 2007*, pages 391–398. ACM, 2007. ISBN 978-1-59593-597-7.

Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In Zoubin Ghahramani, editor, *Machine Learning, Proceedings of the Twenty-Fourth International Conference (ICML 2007), Corvallis, Oregon, USA, June 20-24, 2007*, volume 227 of *ACM International Conference Proceeding Series*, pages 129–136. ACM, 2007. ISBN 978-1-59593-793-3.

David Cossock and Tong Zhang. Statistical analysis of bayes optimal subset ranking. *IEEE Transactions on Information Theory*, 54(11):5140–5154, 2008.

Christopher J. C. Burges, Krysta Marie Svore, Paul N. Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. In Chapelle et al. [2011], pages 25–35.

Xiubo Geng, Tie-Yan Liu, Tao Qin, Andrew Arnold, Hang Li, and Heung-Yeung Shum. Query dependent ranking using k-nearest neighbor. In Myaeng et al. [2008], pages 115–122. ISBN 978-1-60558-164-4.

Irina Matveeva, Chris Burges, Timo Burkard, Andy Laucius, and Leon Wong. High accuracy retrieval with multiple nested ranker. In Efthimiadis et al. [2006], pages 437–444. ISBN 1-59593-369-7.

Koby Crammer and Yoram Singer. Pranking with ranking. In Thomas G. Dietterich, Suzanna Becker, and Zoubin Ghahramani, editors, *Advances in Neural Information Processing Systems 14 [Neural Information Processing Systems: Natural and Synthetic, NIPS 2001, December 3-8, 2001, Vancouver, British Columbia, Canada]*, pages 641–647. MIT Press, 2001.

Amnon Shashua and Anat Levin. Ranking with large margin principle: Two approaches. In Suzanna Becker, Sebastian Thrun, and Klaus Obermayer, editors, *Advances in Neural Information Processing Systems 15 [Neural Information Processing Systems, NIPS 2002, December 9-14, 2002, Vancouver, British Columbia, Canada]*, pages 937–944. MIT Press, 2002. ISBN 0-262-02550-7.

Ping Li, Christopher J. C. Burges, and Qiang Wu. Mcrank: Learning to rank using multiple classification and gradient boosting. In John C. Platt, Daphne Koller, Yoram Singer, and Sam T. Roweis, editors, *Advances in Neural Information Processing Systems 20, Proceedings of the Twenty-First Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 3-6, 2007*. Curran Associates, Inc., 2007.

David Cossock and Tong Zhang. Subset ranking using regression. In Gábor Lugosi and Hans-Ulrich Simon, editors, *Learning Theory, 19th Annual Conference on Learning Theory, COLT 2006, Pittsburgh, PA, USA, June 22-25, 2006, Proceedings*, volume 4005 of *Lecture Notes in Computer Science*, pages 605–619. Springer, 2006. ISBN 3-540-35294-5.

Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, July 23-26, 2002, Edmonton, Alberta, Canada*, pages 133–142. ACM, 2002. ISBN 1-58113-567-X.

Christopher J. C. Burges, Robert Ragno, and Quoc Viet Le. Learning to rank with nonsmooth cost functions. In Bernhard Schölkopf, John C. Platt, and Thomas Hoffman, editors, *Advances in Neural Information Processing Systems 19, Proceedings of the Twentieth Annual Conference on Neural Information Processing Systems, Vancouver, British Columbia, Canada, December 4-7, 2006*, pages 193–200. MIT Press, 2006. ISBN 0-262-19568-2.

Qiang Wu, Christopher J. C. Burges, Krysta Marie Svore, and Jianfeng Gao. Adapting boosting for information retrieval measures. *Information Retrieval*, 13(3):254–270, 2010.

Michael J. Taylor, John Guiver, Stephen Robertson, and Tom Minka. Softrank: optimizing non-smooth rank metrics. In Najork et al. [2008], pages 77–86.

Tao Qin, Tie-Yan Liu, and Hang Li. A general approximation framework for direct optimization of information retrieval measures. *Information Retrieval*, 13(4):375–397, 2010.

Leo Breiman. Random forests. *Machine Learning*, 45(1):5–32, 2001.

Daniel Hasan Dalip, Marcos André Gonçalves, Marco Cristo, and Pável Calado. Exploiting user feedback to learn to rank answers in q&a forums: a case study with stack overflow. In Jones et al. [2013], pages 543–552. ISBN 978-1-4503-2034-4.

Ananth Mohan, Zheng Chen, and Kilian Q. Weinberger. Web-search ranking with initialized gradient boosted regression trees. In Chapelle et al. [2011], pages 77–89.

Hapnes Toba, Syandra Sari, Mirna Adriani, and Ruli Manurung. Contextual approach for paragraph selection in question answering task. In Braschler et al. [2010]. ISBN 978-88-904810-0-0.

Dávid Márk Nemeskey. SZTAKI @ respubliqa 2010. In Braschler et al. [2010]. ISBN 978-88-904810-0-0.

Hitesh Sabnani and Prasenjit Majumder. Question answering system: Retrieving relevant passages. In Braschler et al. [2010]. ISBN 978-88-904810-0-0.

Pamela Forner, Danilo Giampiccolo, Bernardo Magnini, Anselmo Peñas, Álvaro Rodrigo, and Richard F. E. Sutcliffe. Evaluating multilingual question answering systems at CLEF. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, Mike Rosner, and Daniel Tapias, editors, *Proceedings of the International Conference on Language Resources and Evaluation, LREC 2010, 17-23 May 2010, Valletta, Malta*. European Language Resources Association, 2010. ISBN 2-9517408-6-7.

Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In Silva et al. [2007], pages 623–632. ISBN 978-1-59593-803-9.

Meredith Ringel Morris, Jaime Teevan, and Katrina Panovich. A comparison of information seeking using search engines and social networks. In Cohen and Gosling [2010].

Matthew W. Bilotti, Jonathan L. Elsas, Jaime G. Carbonell, and Eric Nyberg. Rank learning for factoid question answering with linguistic and semantic constraints. In Huang et al. [2010], pages 459–468. ISBN 978-1-4503-0099-5.

Adam L. Berger, Rich Caruana, David Cohn, Dayne Freitag, and Vibhu O. Mittal. Bridging the lexical chasm: statistical approaches to answer-finding. In *SIGIR*, pages 192–199, 2000.

Guangyou Zhou, Yang Liu, Fang Liu, Daojian Zeng, and Jun Zhao. Improving question retrieval in community question answering using world knowledge. In Francesca Rossi, editor, *IJCAI 2013, Proceedings of the 23rd International Joint Conference on Artificial Intelligence, Beijing, China, August 3-9, 2013*. IJCAI/AAAI, 2013. ISBN 978-1-57735-633-2.

Wen-tau Yih, Ming-Wei Chang, Christopher Meek, and Andrzej Pastusiak. Question answering using enhanced lexical semantic models. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics, ACL 2013, 4-9 August 2013, Sofia, Bulgaria, Volume 1: Long Papers*, pages 1744–1753. The Association for Computer Linguistics, 2013. ISBN 978-1-937284-50-3.

Xiaoyong Liu, W. Bruce Croft, and Matthew B. Koll. Finding experts in community-based question-answering services. In Otthein Herzog, Hans-Jörg Schek, Norbert Fuhr, Abdur Chowdhury, and Wilfried Teiken, editors, *Proceedings of the 2005 ACM CIKM International Conference on Information and Knowledge Management, Bremen, Germany, October 31 - November 5, 2005*, pages 315–316. ACM, 2005. ISBN 1-59593-140-6.

Nick Craswell, David Hawking, Anne-Marie Vercoustre, and Wilkins Peter. Panoptic expert: Searching for experts not just for documents. In *Ausweb Proceedings*, 2001.

Krisztian Balog, Leif Azzopardi, and Maarten de Rijke. Formal models for expert finding in enterprise corpora. In Efthimiadis et al. [2006], pages 43–50. ISBN 1-59593-369-7.

Pavel Serdyukov and Djoerd Hiemstra. Modeling documents as mixtures of persons for expert finding. In Craig Macdonald, Iadh Ounis, Vassilis Plachouras, Ian Ruthven, and Ryen W. White, editors, *Advances in Information Retrieval , 30th European Conference on IR Research, ECIR 2008, Glasgow, UK, March 30-April 3, 2008. Proceedings*, volume 4956 of *Lecture Notes in Computer Science*, pages 309–320. Springer, 2008. ISBN 978-3-540-78645-0.

Baichuan Li, Irwin King, and Michael R. Lyu. Question routing in community question answering: putting category in its place. In Macdonald et al. [2011], pages 2041–2044. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063885.

Fatemeh Riahi, Zainab Zolaktaf, M. Mahdi Shafiei, and Evangelos E. Milios. Finding expert users in community question answering. In Mille et al. [2012], pages 791–798. ISBN 978-1-4503-1230-1.

Tom Chao Zhou, Michael R. Lyu, and Irwin King. A classification-based approach to question routing in community question answering. In Mille et al. [2012], pages 783–790. ISBN 978-1-4503-1230-1.

Yutaka Kabutoya, Tomoharu Iwata, Hisako Shiohara, and Ko Fujimura. Effective question recommendation based on multiple features for question answering communities. In Cohen and Gosling [2010].

Gideon Dror, Yehuda Koren, Yoelle Maarek, and Idan Szpektor. I want to answer; who has a question?: Yahoo! answers recommender system. In Chid Apté, Joydeep Ghosh, and Padhraic Smyth, editors, *Proceedings of the 17th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Diego, CA, USA, August 21-24, 2011*, pages 1109–1117. ACM, 2011. ISBN 978-1-4503-0813-7. doi: 10.1145/2020408.2020582.

Mingrong Liu, Yicen Liu, and Qing Yang. Predicting best answerers for new questions in community question answering. In Lei Chen, Changjie Tang, Jun Yang, and Yunjun Gao, editors, *Web-Age Information Management, 11th International Conference, WAIM 2010, Jiuzhaigou, China, July 15-17, 2010. Proceedings*, volume 6184 of *Lecture Notes in Computer Science*, pages 127–138. Springer, 2010. ISBN 978-3-642-14245-1. doi: 10.1007/978-3-642-14246-8_15.

Jun Zhang, Mark S. Ackerman, and Lada Adamic. Communitynetsimulator: Using simulations to study online community networks. In Charles Steinfield, Brian T. Pentland, Mark Ackerman, and Noshir Contractor, editors, *Communities and Technologies 2007*, pages 295–321. Springer London, 2007a. ISBN 978-1-84628-904-0.

Jing Zhang, Jie Tang, and Juan-Zi Li. Expert finding in a social network. In Kotagiri Ramamohanarao, P. Radha Krishna, Mukesh K. Mohania, and Ekawit Nantajeewarawat, editors, *Advances in Databases: Concepts, Systems and Applications, 12th International Conference on Database Systems for Advanced Applications, DASFAA 2007, Bangkok, Thailand, April 9-12, 2007, Proceedings*, volume 4443 of *Lecture Notes in Computer Science*, pages 1066–1069. Springer, 2007b. ISBN 978-3-540-71702-7.

Elena Smirnova and Krisztian Balog. A user-oriented model for expert finding. In Paul D. Clough, Colum Foley, Cathal Gurrin, Gareth J. F. Jones, Wessel Kraaij, Hyowon Lee, and Vanessa Murdock, editors, *Advances in Information Retrieval - 33rd European Conference on IR Research, ECIR 2011, Dublin, Ireland, April 18-21, 2011. Proceedings*, volume 6611 of *Lecture Notes in Computer Science*, pages 580–592. Springer, 2011. ISBN 978-3-642-20160-8.

Alessandro Bozzon, Marco Brambilla, Stefano Ceri, Matteo Silvestri, and Giuliano Vesci. Choosing the right crowd: expert finding in social networks. In Giovanna Guerrini and Norman W. Paton, editors, *Joint 2013 EDBT/ICDT Conferences, EDBT '13 Proceedings, Genoa, Italy, March 18-22, 2013*, pages 637–648. ACM, 2013. ISBN 978-1-4503-1597-5.

Theodoros Lappas, Kun Liu, and Evimaria Terzi. Finding a team of experts in social networks. In John F. Elder IV, Françoise Fogelman-Soulié, Peter A. Flach, and Mohammed Javeed Zaki, editors, *Proceedings of the 15th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Paris, France, June 28 - July 1, 2009*, pages 467–476. ACM, 2009. ISBN 978-1-60558-495-9.

Byron Dom, Iris Eiron, Alex Cozzi, and Yi Zhang. Graph-based ranking algorithms for e-mail expertise analysis. In Mohammed Javeed Zaki and Charu C. Aggarwal, editors, *Proceedings of the 8th ACM SIGMOD workshop on Research issues in data mining and knowledge discovery, DMKD 2003, San Diego, California, USA, June 13, 2003*, pages 42–48. ACM, 2003.

Yupeng Fu, Rongjing Xiang, Yiqun Liu, Min Zhang, and Shaoping Ma. Finding experts using social network analysis. In *2007 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2007, 2-5 November 2007, Silicon Valley, CA, USA, Main Conference Proceedings*, pages 77–80. IEEE Computer Society, 2007. ISBN 0-7695-3026-5.

Michael G. Noll, Ching-man Au Yeung, Nicholas Gibbins, Christoph Meinel, and Nigel Shadbolt. Telling experts from spammers: expertise ranking in folksonomies. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *Proceedings of the 32nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009*, pages 612–619. ACM, 2009. ISBN 978-1-60558-483-6.

Pawel Jurczyk and Eugene Agichtein. Discovering authorities in question answer communities by using link analysis. In Silva et al. [2007], pages 919–922. ISBN 978-1-59593-803-9.

Mohamed Bouguessa, Benoît Dumoulin, and Shengrui Wang. Identifying authoritative actors in question-answering forums: the case of yahoo! answers. In Ying Li, Bing Liu, and Sunita Sarawagi, editors, *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, pages 866–874. ACM, 2008. ISBN 978-1-60558-193-4.

Zoltan Gyongyi, Georgia Koutrika, Jan Pedersen, and Hector Garcia-Molina. Questioning Yahoo! Answers. Technical Report 2007-35, Stanford InfoLab, 2007.

Jing Liu, Young-In Song, and Chin-Yew Lin. Competition-based user expertise score estimation. In Wei-Ying Ma, Jian-Yun Nie, Ricardo A. Baeza-Yates, Tat-Seng Chua, and W. Bruce Croft, editors, *Proceeding of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2011, Beijing, China, July 25-29, 2011*, pages 425–434. ACM, 2011. ISBN 978-1-4503-0757-4.

Çigdem Aslay, Neil O'Hare, Luca Maria Aiello, and Alejandro Jaimes. Competition-based networks for expert finding. In Jones et al. [2013], pages 1033–1036. ISBN 978-1-4503-2034-4.

Baichuan Li and Irwin King. Routing questions to appropriate answerers in community question answering services. In Huang et al. [2010], pages 1585–1588. ISBN 978-1-4503-0099-5.

Damon Horowitz and Sepandar D. Kamvar. The anatomy of a large-scale social search engine. In Michael Rappa, Paul Jones, Juliana Freire, and Soumen Chakrabarti, editors, *Proceedings of the 19th International Conference on World Wide Web, WWW 2010, Raleigh, North Carolina, USA, April 26-30, 2010*, pages 431–440. ACM, 2010. ISBN 978-1-60558-799-8.

Shuo Chang and Aditya Pal. Routing questions for collaborative answering in community question answering. In Jon G. Rokne and Christos Faloutsos, editors, *Advances in Social Networks Analysis and Mining 2013, ASONAM '13, Niagara, ON, Canada - August 25 - 29, 2013*, pages 494–501. ACM, 2013. ISBN 978-1-4503-2240-9.

Haiqiang Chen, Huawei Shen, Jin Xiong, Songbo Tan, and Xueqi Cheng. Social network structure behind the mailing lists: ICT-IIIS at TREC 2006 expert finding track. In Voorhees and Buckland [2006].

Lada A. Adamic, Jun Zhang, Eytan Bakshy, and Mark S. Ackerman. Knowledge sharing and yahoo answers: everyone knows something. In Huai et al. [2008], pages 665–674. ISBN 978-1-60558-085-2.

Katrina Panovich, Rob Miller, and David R. Karger. Tie strength in question & answer on social network sites. In Steven E. Poltrock, Carla Simone, Jonathan Grudin, Gloria Mark, and John Riedl, editors, *CSCW '12 Computer Supported Cooperative Work, Seattle, WA, USA, February 11-15, 2012*, pages 1057–1066. ACM, 2012. ISBN 978-1-4503-1086-4.

Lin Chen and Richi Nayak. Expertise analysis in a question answer portal for author ranking. In *2008 IEEE / WIC / ACM International Conference on Web Intelligence, WI 2008, 9-12 December 2008, Sydney, NSW, Australia, Main Conference Proceedings*, pages 134–140. IEEE Computer Society, 2008. ISBN 978-0-7695-3496-1.

Christopher S. Campbell, Paul P. Maglio, Alex Cozzi, and Byron Dom. Expertise identification using email communications. In *Proceedings of the 2003 ACM CIKM International Conference on Information and Knowledge Management, New Orleans, Louisiana, USA, November 2-8, 2003*, pages 528–531. ACM, 2003. ISBN 1-58113-723-0.

Wei-Chen Kao, Duen-Ren Liu, and Shiu-Wen Wang. Expert finding in question-answering websites: a novel hybrid approach. In Sung Y. Shin, Sascha Ossowski, Michael Schumacher, Mathew J. Palakal, and Chih-Cheng Hung, editors, *Proceedings of the 2010 ACM Symposium on Applied Computing (SAC), Sierre, Switzerland, March 22-26, 2010*, pages 867–871. ACM, 2010. ISBN 978-1-60558-639-7.

Hengshu Zhu, Huanhuan Cao, Hui Xiong, Enhong Chen, and Jilei Tian. Towards expert finding by leveraging relevant categories in authority ranking. In Macdonald et al. [2011], pages 2221–2224. ISBN 978-1-4503-0717-8.

Pavel Serdyukov, Henning Rode, and Djoerd Hiemstra. Modeling multi-step relevance propagation for expert finding. In James G. Shanahan, Sihem Amer-Yahia, Ioana Manolescu, Yi Zhang, David A. Evans, Aleksander Kolcz, Key-Sun Choi, and Abdur Chowdhury, editors, *Proceedings of the 17th ACM Conference on Information and Knowledge Management, CIKM 2008, Napa Valley, California, USA, October 26-30, 2008*, pages 1133–1142. ACM, 2008. ISBN 978-1-59593-991-3.

Jiang Bian, Yandong Liu, Ding Zhou, Eugene Agichtein, and Hongyuan Zha. Learning to recognize reliable users and content in social media with coupled mutual reinforcement. In Juan Quemada, Gonzalo León, Yoëlle S. Maarek, and Wolfgang Nejdl, editors, *Proceedings of the 18th International Conference on World Wide Web, WWW 2009, Madrid, Spain, April 20-24, 2009*, pages 51–60. ACM, 2009. ISBN 978-1-60558-487-4.

Bee-Chung Chen, Anirban Dasgupta, Xuanhui Wang, and Jie Yang. Vote calibration in community question-answering systems. In Hersh et al. [2012], pages 781–790. ISBN 978-1-4503-1472-5.

Jiang Bian, Yandong Liu, Eugene Agichtein, and Hongyuan Zha. Finding the right facts in the crowd: factoid question answering over social media. In Huai et al. [2008], pages 467–476. ISBN 978-1-60558-085-2.

Daniel Gildea and Daniel Jurafsky. Automatic labeling of semantic roles. *Computational Linguistics*, 28(3):245–288, 2002.

Alexandru-Lucian Ginsca and Adrian Popescu. User profiling for answer quality assessment in q&a communities. In Jalal Mahmud, James Caverlee, Jeffrey Nichols, John O'Donovan, and Michelle X. Zhou, editors, *Proceedings of the 2013 Workshop on Data-Driven User Behavioral Modelling and Mining from Social Media, DUBMODCIKM 2013 San Francisco, CA, USA, October 28, 2013*, pages 25–28. ACM, 2013. ISBN 978-1-4503-2417-5.

Lawrence Page, Sergey Brin, Rajeev Motwani, and Terry Winograd. The pagerank citation ranking: Bringing order to the web. Technical Report 1999-66, Stanford InfoLab, November 1999. Previous number = SIDL-WP-1999-0120.

Jon M. Kleinberg. Hubs, authorities, and communities. *ACM Computing Surveys*, 31(4es):5, 1999.

F. Maxwell Harper, Daniel Moy, and Joseph A. Konstan. Facts or friends?: distinguishing informational and conversational questions in social q&a sites. In Dan R. Jr. Olsen, Richard B. Arthur, Ken Hinckley, Meredith Ringel Morris, Scott E. Hudson, and Saul Greenberg, editors, *Proceedings of the 27th International Conference on Human Factors in Computing Systems, CHI 2009, Boston, MA, USA, April 4-9, 2009*, pages 759–768. ACM, 2009. ISBN 978-1-60558-246-7.

Felix Hieber and Stefan Riezler. Improved answer ranking in social question-answering portals. In Iván Cantador, Francisco M. Carrero, José Carlos Cortizo, Paolo Rosso, Markus Schedl, and José A. Troyano, editors, *Proceedings of the 3rd International CIKM Workshop on Search and Mining User-Generated Contents, SMUC 2011, Glasgow, United Kingdom, October 28, 2011*, pages 19–26. ACM, 2011. ISBN 978-1-4503-0949-3.

Mark T. Maybury, Oliviero Stock, and Wolfgang Wahlster. Intelligent interactive entertainment grand challenges. *IEEE Intelligent Systems*, 21(5):14–18, 2006.

Michael L. Littman. Review: Computer language games. In T. Anthony Marsland and Ian Frank, editors, *Computers and Games, Second International Conference, CG 2000, Hamamatsu, Japan, October 26-28, 2000, Revised Papers*, volume 2063 of *Lecture Notes in Computer Science*, pages 396–404. Springer, 2000. ISBN 3-540-43080-6.

Christian Bizer, Jens Lehmann, Georgi Kobilarov, Sören Auer, Christian Becker, Richard Cyganiak, and Sebastian Hellmann. Dbpedia - A crystallization point for the web of data. *Journal of Web Semantics*, 7(3):154–165, 2009.

Michael L. Littman, Greg A. Keim, and Noam M. Shazeer. A probabilistic approach to solving crossword puzzles. *Artificial Intelligence*, 134(1-2):23–55, 2002.

Marco Ernandes, Giovanni Angelini, and Marco Gori. Webcrow: A web-based system for crossword solving. In Manuela M. Veloso and Subbarao Kambhampati, editors, *Proceedings, The Twentieth National Conference on Artificial Intelligence and the Seventeenth Innovative Applications of Artificial Intelligence Conference, July 9-13, 2005, Pittsburgh, Pennsylvania, USA*, pages 1412–1417. AAAI Press / The MIT Press, 2005. ISBN 1-57735-236-X.

Giovanni Semeraro, Marco de Gemmis, Pasquale Lops, and Pierpaolo Basile. An artificial player for a language game. *IEEE Intelligent Systems*, 27(5):36–43, 2012.

Shyong K. Lam, David M. Pennock, Dan Cosley, and Steve Lawrence. 1 billion pages = 1 million dollars? mining the web to play "who wants to be a millionaire?". In Christopher Meek and Uffe Kjærulff, editors, *UAI '03, Proceedings of the 19th Conference in Uncertainty in Artificial Intelligence, Acapulco, Mexico, August 7-10 2003*, pages 337–345. Morgan Kaufmann, 2003. ISBN 0-127-05664-5.

Anselmo Peñas, Eduard H. Hovy, Pamela Forner, Álvaro Rodrigo, Richard F. E. Sutcliffe, and Roser Morante. QA4MRE 2011-2013: Overview of question answering for machine reading evaluation. In Pamela Forner, Henning Müller, Roberto Paredes, Paolo Rosso, and Benno Stein, editors, *Information Access Evaluation. Multilinguality, Multimodality, and Visualization - 4th International Conference of the CLEF Initiative, CLEF 2013, Valencia, Spain, September 23-26, 2013. Proceedings*, volume 8138 of *Lecture Notes in Computer Science*, pages 303–320. Springer, 2013. ISBN 978-3-642-40801-4.

Ido Dagan, Oren Glickman, and Bernardo Magnini. The PASCAL recognising textual entailment challenge. In Joaquin Quiñonero Candela, Ido Dagan, Bernardo Magnini, and Florence d'Alché-Buc, editors, *Machine Learning Challenges, Evaluating Predictive Uncertainty, Visual Object Classification and Recognizing Textual Entailment, First PASCAL Machine Learning Challenges Workshop, MLCW 2005, Southampton, UK, April 11-13, 2005, Revised Selected Papers*, volume 3944 of *Lecture Notes in Computer Science*, pages 177–190. Springer, 2005. ISBN 3-540-33427-0.

Álvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. Overview of the answer validation exercise 2008. In Carol Peters, Thomas Deselaers, Nicola Ferro, Julio Gonzalo, Gareth J. F. Jones, Mikko Kurimo, Thomas Mandl, Anselmo Peñas, and Vivien Petras, editors, *Evaluating Systems for Multilingual and Multimodal Information Access, 9th Workshop of the Cross-Language Evaluation Forum, CLEF 2008, Aarhus, Denmark, September 17-19, 2008, Revised Selected Papers*, volume 5706 of *Lecture Notes in Computer Science*, pages 296–313. Springer, 2008. ISBN 978-3-642-04446-5.

Álvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. UNED at answer validation exercise 2007. In Carol Peters, Valentin Jijkoun, Thomas Mandl, Henning Müller, Douglas W. Oard, Anselmo Peñas, Vivien Petras, and Diana Santos, editors, *Advances in Multilingual and Multimodal Information Retrieval, 8th Workshop of the Cross-Language Evaluation Forum, CLEF 2007, Budapest, Hungary, September 19-21, 2007, Revised Selected Papers*, volume 5152 of *Lecture Notes in Computer Science*, pages 404–409. Springer, 2007. ISBN 978-3-540-85759-4.

Eric Breck, John D. Burger, Lisa Ferro, Lynette Hirschman, David House, Marc Light, and Inderjeet Mani. How to evaluate your question answering system every day ... and still get real work done. In *Proceedings of the Second International Conference on Language Resources and Evaluation, LREC 2000, 31 May - June 2, 2000, Athens, Greece*. European Language Resources Association, 2000. ISBN 2-9517408-6-7.

Bernardo Magnini, Matteo Negri, Roberto Prevete, and Hristo Tanev. Is it the right answer? exploiting web redundancy for answer validation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA.*, pages 425–432. ACL, 2002.

Sanda M. Harabagiu and Steven J. Maiorano. Finding answers in large collections of texts: Paragraph indexing + abductive inference. In *Proceedings of the AAAI Fall Symposium on Question Answering*, pages 63–71. AAAI, 1999.

Julio J. Castillo. The contribution of famaf at qa@clef 2008. answer validation exercise. In Peters and Ferro [2008].

Ingo Glöckner. University of hagen at CLEF 2008: Answer validation exercise. In Peters and Ferro [2008].

Kisuh Ahn, Johan Bos, David Kor, Malvina Nissim, Bonnie L. Webber, and James R. Curran. Question answering with QED at TREC 2005. In Voorhees and Buckland [2005].

Christian Bizer. The emerging web of linked data. *IEEE Intelligent Systems*, 24(5):87–92, 2009.

Danica Damljanovic, Milan Agatonovic, and Hamish Cunningham. Freya: An interactive way of querying linked data using natural language. In Raul Garcia-Castro, Dieter Fensel, and Grigoris Antoniou, editors, *The Semantic Web: ESWC 2011 Workshops - ESWC 2011 Workshops, Heraklion, Greece, May 29-30, 2011, Revised Selected Papers*, volume 7117 of *Lecture Notes in Computer Science*, pages 125–138. Springer, 2011. ISBN 978-3-642-25952-4.

Vanessa Lopez, Miriam Fernández, Enrico Motta, and Nico Stieler. Poweraqua: Supporting users in querying and exploring the semantic web. *Semantic Web*, 3(3):249–265, 2012.

E. Cabrio, A. Palmero Aprosio, J. Cojan, B. Magnini, F. Gandon, and A. Lavelli. QAKiS@QALD-2. In C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, and R. Cyganiak, editors, *Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with EWSC 2012*, volume 913, pages 87–95. CEUR Workshop Proceedings, 2012.

Nitish Aggarwal and Paul Buitelaar. A System Description of Natural Language Query Over DBpedia. In C. Unger, P. Cimiano, V. Lopez, E. Motta, P. Buitelaar, and R. Cyganiak, editors, *Proceedings of Interacting with Linked Data (ILD 2012), Workshop Co-Located with ESWC 2012*, volume 913, pages 96–99. CEUR Workshop Proceedings, 2012.

Evgeniy Gabrilovich and Shaul Markovitch. Computing semantic relatedness using wikipedia-based explicit semantic analysis. In Manuela M. Veloso, editor, *IJCAI 2007, Proceedings of the 20th International Joint Conference on Artificial Intelligence, Hyderabad, India, January 6-12, 2007*, pages 1606–1611, 2007.

Vanessa Lopez, Christina Unger, Philipp Cimiano, and Enrico Motta. Evaluating question answering over linked data. *Journal of Web Semantics*, 21:3–13, 2013.

Philipp Cimiano and Michael Minock. Natural language interfaces: What is the problem? - A data-driven quantitative analysis. In Helmut Horacek, Elisabeth Métais, Rafael Muñoz, and Magdalena Wolska, editors, *Natural Language Processing and Information Systems, 14th International Conference on Applications of Natural Language to Information Systems, NLDB 2009, Saarbrücken, Germany, June 24-26, 2009. Revised Papers*, volume 5723 of *Lecture Notes in Computer Science*, pages 192–206. Springer, 2009. ISBN 978-3-642-12549-2.

Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL, 2013. ISBN 978-1-937284-97-8.

Joseph John Jr. Rocchio. Relevance feedback in information retrieval. In Gerard Salton, editor, *The SMART Retrieval System - Experiments in Automated Document Processing*, pages 313–323. Prentice-Hall, Englewood Cliffs, NJ, 1971.

Federico Perea and Justo Puerto. Dynamic programming analysis of the TV game "who wants to be a millionaire?". *European Journal of Operational Research*, 183(2):805–811, 2007.

Royce A. Jr. Singleton and Bruce C. Straits. *Approaches to Social Research*. Oxford University Press, New York, 1993.

Gareth J. F. Jones, Paraic Sheridan, Diane Kelly, Maarten de Rijke, and Tetsuya Sakai, editors. *The 36th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '13, Dublin, Ireland - July 28 - August 01, 2013*, 2013. ACM. ISBN 978-1-4503-2034-4.

Martin Braschler, Donna Harman, and Emanuele Pianta, editors. *CLEF 2010 LABs and Workshops, Notebook Papers, 22-23 September 2010, Padua, Italy*, volume 1176 of *CEUR Workshop Proceedings*, 2010. CEUR-WS.org. ISBN 978-88-904810-0-0.

Marc Najork, Andrei Z. Broder, and Soumen Chakrabarti, editors. *Proceedings of the International Conference on Web Search and Web Data Mining, WSDM 2008, Palo Alto, California, USA, February 11-12, 2008*, 2008. ACM.

Ellen M. Voorhees and Lori P. Buckland, editors. *Proceedings of the Fourteenth Text REtrieval Conference, TREC 2005, Gaithersburg, Maryland, November 15-18, 2005*, volume Special Publication 500-266, 2005. National Institute of Standards and Technology (NIST).

William R. Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson, editors. *The 35th International ACM SIGIR conference on research and development in Information Retrieval, SIGIR '12, Portland, OR, USA, August 12-16, 2012*, 2012. ACM. ISBN 978-1-4503-1472-5.

Erhard W. Hinrichs and Dan Roth, editors. *Proceedings of the 41st Annual Meeting of the Association for Computational Linguistics, 7-12 July 2003, Sapporo Convention Center, Sapporo, Japan*, 2003. ACL.

Sung-Hyon Myaeng, Douglas W. Oard, Fabrizio Sebastiani, Tat-Seng Chua, and Mun-Kew Leong, editors. *Proceedings of the 31st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2008, Singapore, July 20-24, 2008*, 2008. ACM. ISBN 978-1-60558-164-4.

Ellen M. Voorhees and Lori P. Buckland, editors. *Proceedings of the Fifteenth Text REtrieval Conference, TREC 2006, Gaithersburg, Maryland, November 14-17, 2006*, volume Special Publication 500-272, 2006. National Institute of Standards and Technology (NIST).

Proceedings of the International Conference on Language Resources and Evaluation, LREC 2008, 26 May - 1 June 2008, Marrakech, Morocco, 2008. European Language Resources Association.

Efthimis N. Efthimiadis, Susan T. Dumais, David Hawking, and Kalervo Järvelin, editors. *SIGIR 2006: Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, Seattle, Washington, USA, August 6-11, 2006*, 2006. ACM. ISBN 1-59593-369-7.

Olivier Chapelle, Yi Chang, and Tie-Yan Liu, editors. *Proceedings of the Yahoo! Learning to Rank Challenge, held at ICML 2010, Haifa, Israel, June 25, 2010*, volume 14 of *JMLR Proceedings*, 2011. JMLR.org.

Mário J. Silva, Alberto H. F. Laender, Ricardo A. Baeza-Yates, Deborah L. McGuinness, Bjørn Olstad, Øystein Haug Olsen, and André O. Falcão, editors. *Proceedings of the Sixteenth ACM Conference on Information and Knowledge Management, CIKM 2007, Lisbon, Portugal, November 6-10, 2007*, 2007. ACM. ISBN 978-1-59593-803-9.

William W. Cohen and Samuel Gosling, editors. *Proceedings of the Fourth International Conference on Weblogs and Social Media, ICWSM 2010, Washington, DC, USA, May 23-26, 2010*, 2010. The AAAI Press.

Jimmy Huang, Nick Koudas, Gareth J. F. Jones, Xindong Wu, Kevyn Collins-Thompson, and Aijun An, editors. *Proceedings of the 19th ACM Conference on Information and Knowledge Management, CIKM 2010, Toronto, Ontario, Canada, October 26-30, 2010*, 2010. ACM. ISBN 978-1-4503-0099-5.

Craig Macdonald, Iadh Ounis, and Ian Ruthven, editors. *Proceedings of the 20th ACM Conference on Information and Knowledge Management, CIKM 2011, Glasgow, United Kingdom, October 24-28, 2011*, 2011. ACM. ISBN 978-1-4503-0717-8.

Alain Mille, Fabien L. Gandon, Jacques Misselis, Michael Rabinovich, and Steffen Staab, editors. *Proceedings of the 21st World Wide Web Conference, WWW 2012, Lyon, France, April 16-20, 2012 (Companion Volume)*, 2012. ACM. ISBN 978-1-4503-1230-1.

Jinpeng Huai, Robin Chen, Hsiao-Wuen Hon, Yunhao Liu, Wei-Ying Ma, Andrew Tomkins, and Xiaodong Zhang, editors. *Proceedings of the 17th International Conference on World Wide Web, WWW 2008, Beijing, China, April 21-25, 2008*, 2008. ACM. ISBN 978-1-60558-085-2.

Carol Peters and Nicola Ferro, editors. *Working Notes for CLEF 2008 Workshop co-located with the 12th European Conference on Digital Libraries (ECDL 2008) , Aarhus, Denmark, September 17-19, 2008*, volume 1174 of *CEUR Workshop Proceedings*, 2008. CEUR-WS.org.