INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

Answer Selection in Question/Answering

Hugo Patinho Rodrigues

Dissertation for the Degree of Master of
Information Systems and Computer Engineering

Jury

President:	Prof. Doctor Joaquim Armando Pires Jorge
Supervisor:	Prof. Doctor Maria Luísa Torres Ribeiro Marques da Silva Coheur
Member:	Prof. Doctor Bruno Emanuel da Graça Martins

July 2012

Resumo

Esta tese aborda o problema da selecção da resposta na área dos sistemas de Pergunta/Resposta. A selecção da resposta é uma das principais etapas destes sistemas, tendo como objectivo escolher a resposta a devolver com base num conjunto de candidatos. Para tal propomos o AnSelMo (ANSwering SELection MOdule), um módulo de selecção de resposta. A sua abordagem é baseada no contexto onde o candidato de resposta se encontra, medindo as distâncias entre os termos da pergunta e da resposta e usando medidas de similaridade para comparar passagens relacionadas com a pergunta com passagens relacionadas com a resposta. Outra abordagem explorada é baseada em espaços semânticos, usando Análise Semântica Latente.

O AnSelMo foi testado em três cenários distintos: com dados do 'Quem Quer Ser Milionário' (WWBM), o famoso concurso de perguntas de escolha múltipla, no contexto da avaliação conjunta Question Answering for Machine Reading Evaluation (QA4MRE), uma tarefa de compreensão escrita do Cross Language Evaluation Forum, e no Just.Ask, o sistema de Pergunta/Resposta do L^2F. Os resultados para o WWBM ultrapassam o estado da arte, enquanto que para o QA4MRE os resultados são melhores que grande parte dos obtidos pelos sistemas participantes em 2011. Também conseguimos melhorar a exactidão do Just.Ask, através da integração do AnSelMo.

Abstract

This thesis addresses the problem of answer selection in Question Answering (QA) systems. Answer selection is one of the main steps of those systems and has as goal to choose the answer to be returned based on a set of candidate answers. For that, we propose AnSelMo, an ANSwering SELection MOdule. Its approach is based on the context where candidate answers appear, by measuring distances between question and answer terms and by using similarity measures to compare passages related with the question with passages related with the answer. Another approach explored is based in Semantic Spaces, by using Latent Semantic Analysis.

AnSelMo was tested in three different scenarios: in 'Who Wants to Be Millionaire?' (WWBM), the famous contest of multiple-answer questions, in Question Answering for Machine Reading Evaluation (QA4MRE), a Cross Language Evaluation Forum reading comprehension task, and with Just.Ask, the L^2F QA system. Results for WWBM surpass the state of the art, while for QA4MRE results are better than most of the 2011 competing systems' results. We were also able to improve Just.Ask in terms of accuracy, by integrating AnSelMo on it.

Palavras Chave
Keywords

Palavras Chave

Pergunta/Resposta

Selecção de Resposta

Contexto

Proximidade de Palavras

Medidas de Similaridade

Espaços Semânticos

Keywords

Question Answering

Answer Selection

Context

Word Proximity

Similarity Measures

Semantic Spaces

Acknowledgements

First of all, I want to thank my advisor Prof. Luísa Coheur for the opportunity to work with her in such amazing topic, and for providing me, not only a fantastic work environment, but also the motivation and fruitful advises that made this work possible.

I want to thank to all L^2F people as well, specially Ana Mendes, Ricardo Ribeiro and David Matos, for their help, ideas and enlightning discussions.

I am also really grateful to my family, that made me who I am today, for their weariless support.

I could not forget my friends, namely André Carvalho, Cátia Pereira and Diogo Godinho, who were so important throughout these years and were always there when I needed.

And last, but not least, a special thanks to my friend Pedro Mota, who was my companion during this journey, had put up with me when anything gone wrong and gave me his insight in all matters.

Lisboa, July 2012

Hugo Rodrigues

To my grandparents,

António, Maria do Carmo,

Maria de Lurdes and Vasco

Contents

List of Figures

List of Tables

Acronyms

QA Question Answering

WWBM 'Who Wants to Be Millionaire?'

QA4MRE Question Answering for Machine Reading Evaluation

LSA Latent Semantic Analysis

SVD Singular Value Decomposition

MRR Mean Reciprocal Rank

1 Introduction

1.1 Motivation

Question Answering (QA) systems try to answer questions posed in natural language, in contrast to search engines that return a set of related documents given some keywords. This is challenging because natural language is a powerful tool and questions may be formulated in different forms. Also, returning a single and concise answer is a much harder problem than returning a set of more or less related documents.

QA has been studied for some decades, but systems stabilized in the architecture depicted in Figure 1.1.

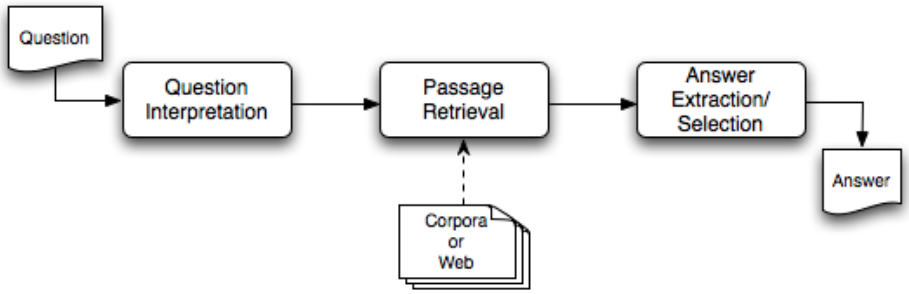

Figure 1.1: QA system typical architecture.

First, the question is processed and interpreted (Question Interpretation step). One or more queries result from this step, which are passed down to the Passage Retrieval step. Here, the system tries to find information related with the question. For this purpose, it can use pre-defined corpora, local databases, or the web. Finally, according to the fetched snippets or documents, the system extracts possible answers and ranks them, selecting one or more as correct answers (Answer Extraction/Selection step).

To exemplify the process, consider the following question: *What is the capital of Portugal?*. The first step could generate queries like `capital Portugal`, `Portugal's capital is` and/or `capital(Portugal, x?)`, depending on the target information source. The following step would retrieve data from corpora, databases or the web, gathering a set of snippets or documents. Continuing with the same example, a system could have as answer candidates, after the answer extraction step, the following set: {*Lisboa*, *Guimarães*, *Lisbon*, *Lisbona*, *Porto*} and it would try to select one or more candidates as correct answers.

The focus of this thesis is in the last task: answer selection. This work aims at exploring this particular step where, given a set of candidate answers – which may or not be related with each other –, we have to choose one or more correct answers.

This problem, as explained, is present in several QA systems, as in Just.Ask [1, 47, 28], the L^2F QA system. Currently, Just.Ask already explores the problem of relating candidates [29], as it joins related candidates into a unique cluster, representing thus a single answer. There is, however, more to exploit. Candidates are selected based on the snippets returned by *Bing*, by extracting candidate answers by their type. However, Just.Ask does not take into consideration the context in which the candidate appears. Recapping the previous example, *Porto* comes from *Porto was chosen, in Portugal, as the European Capital of Culture* while *Lisboa* comes from *Lisboa, capital of Portugal*. Given the context, one could rule out the first candidate in favor of the second one due to the distance between the question keywords, *capital* and *Portugal*, and the candidate answers.

This problem can also be found in other scenarios in the Natural Language Processing (NLP) field. For example, Question Answering for Machine Reading Evaluation (QA4MRE), a task introduced in Cross Language Evaluation Forum (CLEF) in 2011 [34], aims at evaluating a system's reading capability through multiple-answer questions. Briefly, given a text and some questions about it, competing systems have to choose the correct answer to those questions, among five candidates, showing in this way their level of "comprehension" of the text. The information needed to correctly choose between the different questions can be found in the given texts. The task also provides a Background Collection, which consists in a collection of

documents that can help systems answer the questions.

Another similar scenario, already studied by the NLP community, is 'Who Wants to Be Millionaire?' (WWBM), the famous multiple-answer question contest. In this case, contestants face different questions, with four hypothesis of answer.

These two tasks can be distinguished from the answer selection problem in QA for two reasons: (1) they always have a correct candidate answer; and (2) the candidate answers are not related, as opposite to the candidate answers extracted, for instance, by Just.Ask. Thus, these scenarios can be seen as a subproblem of the answer selection problem, as there are only up to five candidates, non-related, and where one and only one is the correct answer. However, these two are also distinct scenarios, due to their nature and complexity (QA4MRE questions tend to be trickier, as it has a Machine Reading (MR) task associated with it).

1.2 Goals

In summary, this dissertation aims at:

- Presenting the current state of the art on answer selection;
- Proposing techniques that consider the context of candidate answers to select them;
- Developing an answer selection module based on those techniques;
- Applying those techniques to Just.Ask, WWBM and QA4MRE scenarios and evaluate their impact.

1.3 Approach

We follow two different approaches to the answer selection task: a Lexical Distance-based approach and a Semantic Spaces approach. The first is based on two works from the beginning of the century, applied to WWBM, that we recover, modify and extend for this purpose. Both

are directly related with the context of the candidate answer, and use different techniques to measure it. The second one is based on semantic spaces, and tries a different approach to the problem. The idea is to discover latent topics from documents, and associate the question and candidate answers to those topics, distinguishing them in the process.

1.4 Contributions

The work of this thesis led to the following contributions:

- A survey and analysis of the current state of the art on answer selection;
- The development of a set of techniques that consider the context of candidate answers, from where they are extracted, in order to select them;
- An answer selection module, AnSelMo, based on those techniques;
- The study of applying AnSelMo to three different scenarios: WWBM, QA4MRE and Just.Ask.

This work also resulted in a paper entitled *2B$ – Testing Past Algorithms in Nowadays Web*, accepted in TSD 2012, the 15th International Conference on Text, Speech and Dialogue, that will take place in Brno, Czech Republic, in September 3–7 2012.

This work also allowed L^2F to participate in the 2012 QA4MRE track.

1.5 Structure of this Document

This document is structured as follows: in Chapter 2 we present related word; Chapter 3 describes our system and its techniques, followed by the experiments performed, detailed in Chapter 4. Finally, in Chapter 5, we present the conclusions of this work and point to some future work.

2 Related Work

In this chapter we present the related work. We first analyze the problem of answer selection in QA systems, followed by two works applied to the WWBM problem (Section 2.2). In Section 2.3 we describe the participating systems in 2011 QA4MRE track, in Section 2.4 we present Just.Ask and in Section 2.5 we give some insight into Latent Semantic Analysis (LSA). Finally, we describe some information sources and end with a summary of the chapter.

2.1 Answer Selection in Question Answering

In this section, divided in subsections, are described the different approaches studied to answer selection.

2.1.1 Pattern-based approach

Some systems are based in patterns, and, thus, sometimes it is hard to distinguish the extraction step from the answer selection step. Gonzalez et al. [15] developed a system to participate at CLEF2006 that uses regular expressions to extract candidate answers from the passages gathered before. These patterns are designed to retrieve the answer depending of the question type. Answers are then categorized with a set of 17 attributes. These attributes may represent the lenght of the questions, the similarity between the question and the candidate answer (by the number of common lemmas, named entities, etc.) and the redundancy of the candidate answer in all passages. These are used by a Naïve Bayes classifier, which selects the candidate answer with most probability of being the correct one. CINDI_QA [16], from CINDI group, also participated at CLEF (2007) and uses a similar approach. Answers are

extracted from what they call of templates, which are no more than pre-defined lexical patterns. Raposa [45], a QA system for the Portuguese language, also uses patterns to extract the candidate answers. Although the authors refer to the patterns as "a set of context evaluation rules", they are no more than strict patterns that are applied depending of the question type. There are also other examples of systems that use patterns to extract candidate answers, like Kupiec [21], Soubbotin [49] and Fleischman et al. [13]. Hearst [18] also used patterns, but in a different task: to automatically acquire WordNet [30] relations [17]. Actually, Fleischman et al. [13] refers that its patterns occur 40 times often than those employed by Hearst. Joho and Sanderson [20] also based their work on Hearst to answer definition questions by applying similar patterns. An example of those patterns could be '`<name>, such as <hyponym`$_1$`> or <hyponym`$_2$`>`', that identifies hyponymies of a given word.

Patterns are also used by Ravichandran and Hovy [39]. However, the applied patterns are automatically extracted (in fact, great part of the patterns applied by Hearst are discovered automatically too). To do this, the authors learn the patterns from a seed. A seed is a sample containing the question term (for example 'Mozart') and the correct answer ('1756'), for the given question category (in this case 'birthyear'). The pair is submitted to a search engine and the top N results containing both terms are used to extract a pattern able to match the question term and answer. Those patterns are then generalized, by swapping the question term and answer by the <NAME> and <ANSWER> tags, respectively. In this example, patterns like '`born in <ANSWER> , <NAME>`', '`<NAME> was born on <ANSWER>`', '`<NAME> ( <ANSWER> -`' and '`<NAME> ( <ANSWER> - )`' could be extracted. The process is repeated with other seed pairs, learning thus more patterns and refining the existing ones. After this, new queries are created from the seeds with only the question terms (without the answer this time) and the obtained patterns are used to extract the answers. The answers retrieved with the patterns are then compared with the expected answer. The ratio of correctly extracted answers becomes the precision of the pattern that originated such answer. The values found represent a probability of each pattern to find the correct answer and are used later to decide what candidate answers are returned as the correct ones.

Sun et al. [53] use a similar strategy to extract answers: a statistical approach for pattern matching. However, unlike the previous approach, which relies on patterns' confidence, their system calculates the similarity between the question and the snippet containing the candidate answer. The system allows two different matchings: a strict one, where the stems from both sentences should be the same, and a flexible one, which relates words with the help of WordNet. Matches are weighed and summed up, totalizing a score for the given answer.

Priberam's QA system [7] also uses patterns to extract candidate answers. However, it uses an answer validation module to ensure the correctness of the answers. This is done by applying some sanity checking techniques, such as matching of named entities (similar to the previous matching technique). The idea is to match the named entities present in the snippet containing the candidate answer with those in the question. In case of failure, the candidate is discarded. Rodrigo et al. [41] use a similar approach at 2007 Answer Validation Exercise (AVE), on CLEF. In AVE, systems are required to decide if an answer from a QA system is correct or not, labeling it as `Validated`, `Selected` or `Rejected` [42].

2.1.2 Knowledge-based approach

Other strategy sometimes applied is known as model-based answer selection and can be seen as a knowledge-based approach. The idea suggested by Sinha and Narayanan [48] is to model the relations between events, entities and their properties, reasoning then about them to reach the correct answer. The system is able to parse documents and extract properties, relating them with those from the question. As example, consider the question *Does Pakistan possess the technological infrastructure to produce biological weapons?*. The system extracts two relations, `possess(Pakistan, technologicalInfrastructure)` and `produce(Pakistan, biological Weapons)`, and will later consult an ontology, searching for the preconditions necessary to a country produce biological weapons, what expertise is needed and if Pakistan possesses it.

2.1.3 Noisy-Channel-based approach

The two techniques presented before (pattern-based and knowledge-based) are described by Echihabi et al. [11], as well as a noisy-channel-based approach to answer selection. Noisy-channel models are commonly used for error correction. Given a word, which may have some letters scrambled (or missing or even some extra letters), the calculated model shows the probability of the given word correspond to some other word (belonging to an allowed lexicon). The authors' proposal is to consider sentences instead of words and, thus, words instead of letters. Given a sentence obtained from an information retrieval system one can calculate what is the probability of transforming that sentence into the original question. The trained model gives such probability and the answer is extracted from the snippet which probability is higher.

2.1.4 N-grams approach

Aranea [26, 25] and OpenEphyra [46] use similar strategies between them. Aranea generates all n-grams of terms, from unigrams to tetragrams, from retrieved passages. These n-grams have an initial score, depending from which query they are from, and are considered the candidate answers. Identically, candidate answers extracted by patterns in OpenEphyra are given an initial score. Those equal candidates (or n-grams in case of Aranea) are then merged, summing up their scores. This process although gives more weight to the popular answer is able to filter some candidates resulting from malformed documents and incorrect information, as they should be rarer. Even that the previous step allows some filtering, there is a filtering step afterwards. Candidates are filtered considering some heuristics such as the candidates matching the expected answer type.

Because Aranea is dealing with n-grams, another combining step is performed. The remaining candidates are combined now taking into account the intersection of the various n-grams. If a candidate answer is a portion of another one, then the two are merged together, again summing their scores. Longer answers are preferred, counteracting this way the weight given

by the previous combination step to shorter n-grams. Finally, Aranea multiplies candidate scores by a factor that considers the terms present in the answer.

The authors of Aranea also refer some limitations of their system. Questions like *Who was the first president of Portugal?* could lead to the actual president. The reason is that Aranea does not recognize temporal expressions. Due to its approach, words like *first* or *last* are considered unimportant. We found no evidence showing that OpenEphyra would not be prone to the same problems.

2.1.5 Large-Scale Question Answering

DeepQA (or Watson) [12], unlike the previous examples, relies on massive parallelism and many expertise, i.e., it is able to run different techniques, often heavy and long taking, at the same time and much quicker. Thus, a number of different algorithms are applied. None of them is predominant, which means that the combination of them turns it possible to find the correct answer to almost all questions and no matter how they are posed. DeepQA employs over 2500 3.5GHz POWER7 cores and 16 Terabytes of RAM, being able to process 500 gigabytes per second. As we can see, Watson is a mega-project from IBM: this power enables the computation of dozens of algorithms, which is unthinkable for us, since we do not possess such resources. Nevertheless, the process deserves mention.

Candidate answers are gathered from different information sources and suffer a soft filtering. These filters all but close to 100 candidates. The filtering is done with some 'lightweight' techniques, such as the candidate answer matching the question type. After this, there is a scoring step. As mentioned, DeepQA allows various techniques. About 50 scoring methods are used, from counting categorical features to information from triple stores. Each scorer may be focused on one or more features and work independently, making it easier to add more scorers in the future.

Because it would be hard to compare directly scores from different methods, DeepQA needs to normalize the candidates. As seen in the previous systems, Watson also merges similar

candidates. It uses candidate matching and coreference resolution, among others, to identify related candidates. In the end, candidates are ranked and the confidence in each one is calculated (this confidence is part of the strategy to play *Jeopardy!*, which is out of the scope of this thesis).

2.2 'Who Wants to Be Millionaire?'

As mentioned before, WWBM is one of this thesis' scenarios. It is composed by a set of multiple-answer questions, each with four candidate answers – where one is the correct answer. Two approaches were found in the literature, and are presented in this section.

2.2.1 The approach by Clarke et al.

To our knowledge, the first computational attempt to solve WWBM is the one described by Clarke et al. [9], where the authors present an approach to the QA task (first used at Text REtrieval Conference (TREC) [8]) and, ultimately, on WWBM. The main idea is to retrieve *extents* and score them accordingly to the likelihood of them containing the answer. Given a document D, which is seen as a set of terms ordered sequentially, one can have a subsequence defined by a pair (u, v), referred as extent. This extent represents a passage, starting at term t_u and ending at term t_v. These extents are created based on queries (Q), generated from the original question. The authors established also two concepts: *satisfaction* and *cover*. An extent is said to satisfy a term set $T \subseteq Q$ if the subsequence that the extent represents contains at least once each term in T. If there is no minor subsequence that satisfies T, then the given extent is considered to be a cover for T. T is no less than the power set of $Q, \mathcal{P}(Q)$.

The extents defined as covers are then scored. The score reflects the possibility of finding the answer in the extent or in its neighborhood. For each term $t_i \in T$, there is an associated probability p_t, which corresponds to the likelihood of it appearing in any location in the document. It is made the assumption that a document is a set of independent terms. Although the assumption does not hold in general, as some terms appear more frequently together, it is

often a fair approximation given a large corpus. The probability p_t can be roughly estimated from the frequency of the term in the document, $p_t = f_t/N$, with N being the length of the document, that is, the number of terms present. The probability of a given term being present on an extent (u, v), with length $l = v - u + 1$, can be thus defined as:

$$\begin{aligned} P(t,l) &= 1 - \overline{P(t,l)} \\ &= 1 - \big(\overline{P(t,u)} \times \overline{P(t,u_{+1})} \times \cdots \times \overline{P(t,v)}\big) \\ &= 1 - \big((1 - P(t,u)) \times (1 - P(t,u_{+1})) \times \cdots \times (1 - P(t,v))\big) \end{aligned}$$

Given the assumption of independence of a term appearing in any location with probability p_t, we can resume the equation to:

$$\begin{aligned} P(t,l) &= 1 - (1 - p_t)^l \\ &= 1 - (1 - lp_t + xp_t^2 - yp_t^3 + \cdots \pm p_t^l) \\ &\simeq 1 - \big(1 - lp_t + O(p_t^2)\big) \end{aligned}$$

The values l, x and y can be extracted from the first three values of the l^{th} line from the Pascal Triangle. As p_t is necessarily lesser than 1, the greatest the power it has, the smaller it becomes. Therefore, one can look only at p_t^2 as the worst case (the asymptotic limit), leaving the first two terms of the expansion and contracting the remaining. After applying the minus signal, we end up with a more friendly value (approximately):

$$P(t,l) \simeq lp_t \tag{2.1}$$

After finding the value for $P(t,l)$ we can now use it to calculate the probability of all terms in T appear in the extent (u, v):

$$\begin{aligned} P(T,l) &= \prod_{t \in T} P(t,l) \\ &\simeq \prod_{t \in T} lp_t \end{aligned}$$

$$= l^{|T|} \prod_{t \in T} p_t$$
$$= l^{|T|} \prod_{t \in T} f_t / N \tag{2.2}$$

After this we are able to score the extents. The authors calculate the score of each extent by calculating the amount of self-information it contains by having all terms $T \subset Q$. Self-information measures the information associated with the occurrence of a given event and its outcome. The smaller the probability of an event, the greater the self-information associated with the occurrence of that event. This measure is additive and positive. Thus, if we have an event which is the composition of other two independent events, the self-information associated is the sum of self-information of those two events. The formula for self-information is:

$$I(\omega_n) = \log\left(\frac{1}{P(\omega_n)}\right) = -\log P(\omega_n), \tag{2.3}$$

where ω_n denotes the outcome of the event with probability $P(\omega_n)$.

The score of an extent (u, v) is no more than the self-information contained by (T, l), which can be calculated based on Equation 2.2 and substituting it in Equation 2.3. Because we are assuming the independence of each $P(t, l)$, we can simplify the equation:

$$\begin{aligned} I\big((T,l)\big) &= -\log P(T,l) \\ &= -\log(l^{|T|} \prod_{t \in T} p_t) \\ &= -\log(l^{|T|}) - \log(\prod_{t \in T} p_t) \\ &= -|T| \log l - \sum_{t \in T} \log(p_t) \qquad (2.4) \\ &= \sum_{t \in T} \log(1/p_t) - |T| \log(l) \\ &= \sum_{t \in T} \log(N/f_t) - |T| \log(l) \qquad (2.5) \end{aligned}$$

Equation 2.4 gives a higher score to snippets with lower probability of occurrence. These

higher values do not imply necessarily a greater likelihood of the answer being present in the extent or its proximity, but the authors found empirically that this relation holds. Just with this passage retrieval technique, at TREC-8, the system found about 63% of correct answers.

Initially, as mentioned before, this strategy was applied to TREC. Here, an answer is a passage, not a term. Thus, a passage centered in the extent was returned. To consider single terms as candidates, instead of complete snippets, the score is based on a simplification of Equation 2.5. The second part of the equation is reduced to 0, since l is now 1. The first one loses the sum for the same reason (only one term). The scoring for a term t is then:

$$\begin{aligned} w_t &= c_t I\big((t,1)\big) \\ &= c_t \log(N/f_t), \end{aligned} \tag{2.6}$$

where c_t is the redundancy component, representing the number of occurrences of t in the different passages.

Finally, the authors tried to use these techniques for multiple-answer questions, from the WWBM contest. Because there is a set of possible answers, the passage retrieval module was modified to take that into account. For this, the scoring process also cares about the presence of one of the answers in the passages (still using Equation 2.4). The passages are then ranked by votes (one for each answer present in the snippet). The accuracy accomplished for a 108-question test was 70%.

2.2.2 The approach by Lam et al.

In 2003, Lam et al. [22] took another approach to WWBM. The strategy is simple, yet with good results. The authors used the web as their only resource, and reported three different techniques to this problem, with accuracies ranging from 50 to 75%.

The first technique comes out directly from the problem formulation. Having the possible answers (and, of course, the correct one among them), is it easier to confirm each one of them instead of searching just for the correct one? Relying on web, is it possible to use search

engines to answer the questions this way? The authors established their baseline choosing as correct answer the one with more results retrieved by querying a search engine with a query in the form 'answer . questionModified'. The term 'questionModified' refers to the question filtered of stopwords. Also, for questions with a *not* presented, like *Which of these plays is not written by Shakespeare? A: Hamlet B: Othello C: Romeo and Juliet D: Cats*, the chosen answer is the one with less results (an inversion due to the presence of *not*). Using *Google* the accuracy was set in about 50%. Some modifications, as enclosing answers by quotes in the queries, allowed the system to reach a percentage of correct answers of 60%.

Another technique employed by the authors considers the position of the answer in a given document, when compared with other words present in the question. The rational for this heuristic is the observation that often answers appear in documents close to question words. It is intended to weigh the distances, of a maximum radius, so that documents with too many references to an answer but not to the corresponding question words are worth less. The first ten pages returned by the queries are used in the algorithm presented in Algorithm 1. The parameter 'documentSplit' represents an array where each position is a term in the document.

Algorithm 1 Pseudocode for a scoring algorithm, giving more weight to answers near question words, within radius words.

```
1: DistanceScore(documentSplit, questWords, ansWords, radius)
2: score, ansFoundWords = 0
3: for i = 1 to ||documentSplit|| do
4:    if documentSplited[i] ∈ ansWords then
5:       ansFountWords += 1
6:       for j = (i−radius) to (i+radius) do
7:          if documentSplited[j] ∈ questWords then
8:             score += (radius −|i − j|)/radius
9:          end if
10:      end for
11:   end if
12: end for
13: if ansFoundWords == 0 then
14:    return 0
15: else
16:    return score/ansFoundWords
17: end if
```

Figure 2.1, extracted from Lam et al. [22, Figure 1], depicts the accuracies for three runs of

90-questions for different values of radius. The graph also shows the accuracy of the baseline approach.

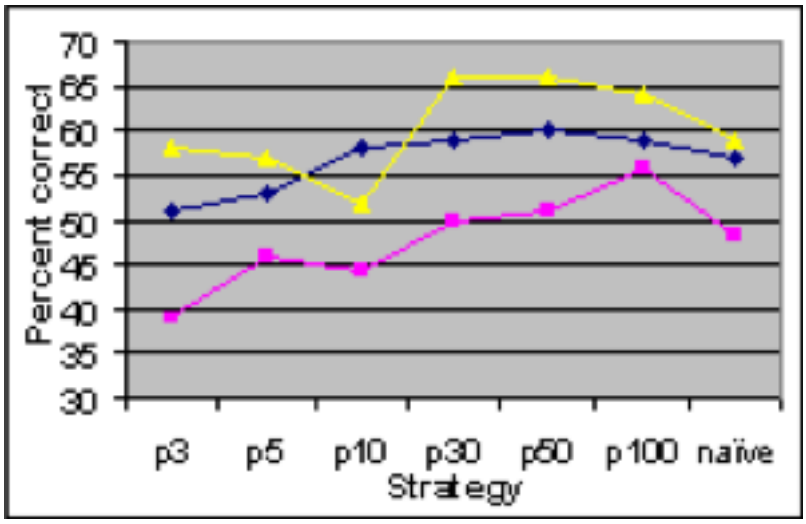

Figure 2.1: Algorithm 1 performance for three 90-questions runs, with different radius. The last point corresponds to the baseline approach.

Inspecting the figure we can see that the optimal value for radius is not trivial to derive, but for higher values the algorithm performs better than simply counting the results for each possible answer.

Greater accuracies (70-75%) were accomplished by combining different runs and different search engines.

2.3 Question Answering for Machine Reading Evaluation

QA4MRE is another of the scenarios proposed in this thesis. Recall that in this task systems must answer a set of multiple-answer questions, related with a document (a TED talk transcription). There are five candidate answers and, as in WWBM, candidate answers are not related, being one of them the correct answer.

In 2011, about twelve systems participated in this challenge, from which ten in the English task. From the twelve, there are only eight working notes available [34]. For all submitted

runs (43 for english, 11 for german and 9 for romanian), nearly half reported score below the baseline [34, Table 11], which is set on 0.2 in c@1, meaning random guesses to all questions. The winning system [33] achieved a 48.3% accuracy (0.57 considering the c@1 measure).

The systems presented in 2011 can be divided in three categories: the ones using Information Retrieval (IR) techniques, those which view QA4MRE as a QA problem and, finally, those applying logic-based strategies.

In the last group is the system proposed by Babych et al. [3], for German. It parses the input text, representing it in a dependency graph. From it, the system uses a set of rules as Hearst [18] did to extract hyponym and other relations. Finally, all sentences and rules form the previous step are transformed into logical formulae and, given the candidate answer C, the input text T, and the Background Collection B, the system tries to infer if $(T \wedge B) \vdash C$. Similarly, Glöckner et al. [14], also for the German task, take each candidate answer and, together with the question, build an hypothesis, which should be proved from the reading test and the Background Collection.

Verberne [54] submitted a system based on IR techniques. After data preparation, including coreference resolution (which the last mentioned system also does), sentences from input text are extended with sentences from the Background Collection. This is done with the BM25 ranking function, which ranks passages from Background Collection accordingly to their similarity to input text. Also, questions are extended with facts retrieved from the Background Collection. For this, the system uses a set of rules, which extract facts that establish a relation between a subject and an object. Then, if a question contains any of the subjects and/or object, the respective fact is added, extending this way the question. These two strategies are optional, and the system behaves better when using only the first one. Independently of which one is used, the system then uses again BM25 to measure the similarity between questions (extended or not) and the passages from input text (extended or not), ranking them in the process. Finally, for each answer, the system uses again BM25 to score them against the selected fragments (extended or not), resulting in the final answer.

Iftene et al. [19] also applied a simple IR technique, which consisted in using the Lucene indexer[1] to the reading test texts and Background Collection. The built index is then queried using the questions, creating this way another index, built with the retrieved passages/documents. Then, based on this new index, answers are used as queries in a new retrieval step. The relevance scores from each retrieval step are then used to compute a final score for each answer. To transform questions and answers into queries, the authors use Named Entity (NE) Recognition (NER), lemmatization and stopword elimination, giving more weight to NEs when building the indexes.

Saias and Quaresma [44] also used an approach based on Lucene. With the texts and Background Collection indexed, the system uses WordNet [30] to check for synonyms and uses the answers as queries, filtered of stopwords. Then, for each answer, the system checks two rules: the first one depends on the question classification, which, according to the authors, did not get any results. Each classification is associated with a pattern, with which was tried to extract the answer. If it is unsuccessful, then the second rule applies. This second rule measures the distance between the key elements (i.e. object of the question) and the answer. The answer selection follows a set of if-else clauses regarding the two rules. The conditions are the rules that fired, how many times they did fire, or the minimal distance calculated.

Cao et al. [6] presented a strategy that simulates a strategy applied by people when learning a new language and answering reading tests. According to the authors, people will first locate the passages related with the questions, searching them by NEs. Thus, their system performs NER to find related passages afterwards, by comparing the NEs between the question and passages. We should note that the texts are preprocessed to resolve anaphoras. The second and third steps are hardly dissociable: the system uses a lexical chain to score each hypothesis, selecting then the one with greater score. The lexical chain was based on previous work [31]. The idea is to relate terms by WordNet relations, such as synonym, hypernym and gloss. The terms may be directly related (a V relation – the letter is associated with how many intermediate terms exist) or indirectly, having one or more terms between (W, VW and

[1] http://lucene.apache.org/

WW relations). Each kind of relation has a weight associated, which contributes to the final score. Unfortunately, the authors' description is not clear on how answers and passages are compared.

Martinez-Romo and Araujo [27] used a completely different approach. Although the Background Collection is also preprocessed and indexed by Lucene, the system links all nouns (proper and common) and verbs within a given document, establishing a co-ocurrence graph. This means that the terms appearing in a given document are related under the same topic. After this, the system uses WalkTrap [37] to automatically discover communities. These communities are basically clusters that gather terms belonging to the same topic. Then, each question is assigned to a community based on their similarity. Following this, each answer is also assigned to a community; the selected answer is the one with greater similarity to the question context (i.e., community). We should note that the authors do not specify what are the similarity measures used to compare questions with communities, answers with communities, and the communities themselves.

Pakray et al. [33] developed a system that uses two different strategies: an Answer Validation (AV) approach and a QA approach. The best results were accomplished by using the system as an hybrid between the two. The AV module is based on Textual Entailments (TEs): for each answer, for a given question, an hypothesis H is generated, according to a set of patterns. These are then used to retrieve passages from the texts, which are indexed with Lucene. The topmost sentence, T, is paired with the corresponding hypothesis, resulting in the pair T-H. These pairs are then processed by a pipeline of different strategies to determine if they are TEs. Among these strategies are the comparison of NEs, the number of co-occurring unigrams, bigrams and skip-bigrams between T and H, and the matching of question and answer types. Finally, the pair with greatest score from all strategies is chosen and the associated answer is considered as the correct answer. The QA module starts by doing similar tasks. Following some rules, each question is transformed into a pattern, where the wh-word is substituted by one of the candidate answers. From these patterns are also extracted stopwords, creating a keyword list. Then, each pattern is compared against each sentence from the documents. If

they do not match, the same is done between the respective keyword list and the sentences. Whichever matches, a score is assigned. Finally, the answer associated with the sentence with greatest score is chosen as the correct answer. It is important to note that anaphora is resolved in each document. For this, NER is performed, in order to help in this task. First person personal pronouns are substituted with the document's author; however, in direct speech, these pronouns refer to the last NE, making the swap accordingly. For third person pronouns the system uses similar rules to detect the referring NE, depending if it is direct or indirect speech.

2.4 Just.Ask

Just.Ask [1, 47, 28] is a QA system developed by the L^2F group and it is the third scenario of this thesis.

Just.Ask receives a question and starts by analyzing it. For that it uses tools like tokenizers and syntactic analyzers. The question headword is extracted and the question is classified according to the two-layer taxonomy of Li and Roth [24]. This information is passed down to the passage retrieval module. Just.Ask uses different information sources and, thus, different query formulations, depending of their purpose.

The relevant passages retrieved in the previous step are the input for the answer extraction and selection step. Just.Ask uses a set of different approaches, each one for a subset of questions. For NUMERIC and ABBREVIATION type questions, Just.Ask utilizes a set of patterns to extract candidate answers. However, questions of type HUMAN:INDIVIDUAL require a more flexible approach, because names may appear in different formats and are therefore difficult to express in patterns. For this Just.Ask uses a machine learning-based named entity recognizer. There is still a third kind of questions, the *type of* questions. These questions ask for hyponyms of the questions headword. For instance, in the question *Which animal is the fastest?*, the answer we are looking for is not an individual, but a type of animal: cheetah, which is an hyponym of animal. Just.Ask uses WordNet [30] at run time, extracting such relations to

answer these type of questions.

After candidate answers are extracted, Just.Ask chooses one to be returned as final answer. Candidates go through three steps: normalization, clustering and filtering.

For the first step, Just.Ask normalizes candidate answers that correspond to the same entity. The idea is to try to eliminate the variation of answers. This is accomplished by transforming candidates to a unique representation. For example, numbers (numeric or textual versions) are all transformed into a number with one decimal place.

After normalization, the clustering step is performed. The goal is to merge candidate answers representing the same entity into a unique cluster. Just.Ask uses two distance measures to determine the similarity between candidates: Overlap distance and Levenshtein distance [23]. The first is defined as follows:

$$\text{Overlap}(X, Y) = 1 - \frac{|X \cap Y|}{\min(|X|, |Y|)}, \tag{2.7}$$

where $|X \cap Y|$ is the number of tokens present in both candidate answers X and Y, and $\min(|X|, |Y|)$ is the size of the smallest candidate.

Initially each candidate answer has its own cluster. After each iteration, the two closest clusters are merged together. The distance between clusters is defined by the minimum distance between any member of each cluster. When the process halts, the longest candidate of each cluster is chosen as the representative answer of that cluster.

Finally, a filtering step occurs. Here, clusters with any member present in the original question are discarded. After filtering, the system chooses the cluster with higher score and returns its representative answer. The score of a given cluster is simply the sum of the scores of each member, which are attributed during the answer extraction phase.

2.5 Latent Semantic Analysis

LSA is a mathematical technique first applied to IR [10] that has been also applied to different areas, such as summarization [40, 32]. LSA can help to identify latent topics from documents, even without knowing them *apriori*.

LSA is a vector-space approach, based on Singular Value Decomposition (SVD) to reduce the dimensionionality from the original matrix A. This matrix is usually a *term by document* matrix, but sentences instead of entire documents may also be used. The matrix has dimensions $T \times N$, with T being the total number of terms and N the total number of documents. The matrix is filled with the scores of each term in the given document. These scores usually have a local and global component (for instance, `tf.idf` score may be used), but only local weights such as term frequencies or other features may be used. From SVD application to matrix A results a decomposition into three matrices:

$$A = U\Sigma V^T,$$

with U being a $T \times m$ matrix, Σ a $m \times m$ diagonal matrix, whose elements are non-negative singular values sorted in descending order and V a $m \times N$ matrix. Figure 2.2 depicts the transformation.

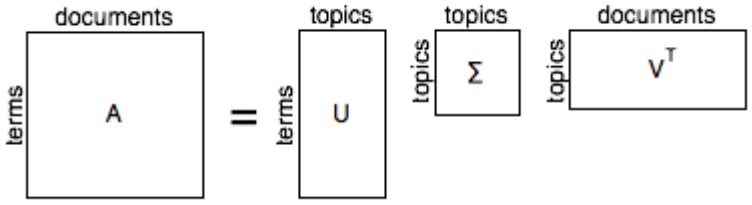

Figure 2.2: Application of SVD to matrix A.

We can see that one can now define terms by topics (or concepts), and also topics by document. This allows another view on the documents, For example, in summarization one can extract only a subset of topics from a document and then select passages from that topic. The singular

values from Σ matrix follow a Zipf distribution, meaning that the top-n topics are the most relevant, while the remaining have approximately the same importance.

This decomposition allows different analysis. For example, one can compare two documents by evaluating their similarity. This is done by comparing their vectors (which are normalized due to SVD), applying, for instance, a cosine function between them.

There are similar approaches, using the same paradigm (semantic spaces and dimension reduction), but that rely in probabilities instead of frequencies. These are called Probabilistic Topic Models, and two examples are pLSA (LSA with a probabilistic component) and LDA (Latent Dirichlet Allocation), which differs by adding a Dirichlet prior on the topic distribution [50].

2.6 Information Sources

Although we are focused in answer selection, such task is not possible without performing passage retrieval. In fact, if the correct answer has not been extracted before, there is no hope for the system to answer correctly the question later. QA systems need, thus, information sources. Information sources can be classified into two different groups: non-structured data and (semi-)structured data. The first refers to textual documents on the World Wide Web (WWW). The latter is related to databases and other platforms where information can be easily extracted. The two types are discussed in detail in the following subsections.

2.6.1 World Wide Web as an information source

The web is one of the most accessible information sources and is an example of non-structured data, in the sense that the data is contained in free text, that is, the content of the web is not structured, although web pages are structured in HTML. One significant advantage of the web is that it has redundant data. Most systems that use the web rely on this to support their answers. The assumption is that correct answers appear often in the web, while less accurate answers appear fewer times. However, one of the main limitations of this strategy is that a system may end up answering not the correct answer, but the most popular one, as

WWW contains much noise due to being wide-spread and of free-usage (anyone can create a blog nowadays, writing there whatever (s)he wants to).

Search engines are, no doubtfully, an important component of any system that uses the web. They permit, easily, to fetch a set of (more or less) related documents, given a number of keywords. Typical examples are *Google* or *Yahoo*. People are used to input some terms in their browser and find quickly the intended information, almost always in the first page of returned results. Although humans are pretty accurate doing this, the task of extracting an answer from the snippets returned by the search engine is much harder for a computer. Concretely, a system may be confused by surrounding terms, misleading the answer extraction module. For instance, systems using patterns (like those presented in Section 2.1.1) may not extract the correct answer due to this fact. Thus, the snippets retrieved should contain the least possible noise, to further facilitate the extraction.

As examples of systems that use the web we have the previously mentioned Just.Ask [1, 47, 28] using *Bing*, Aranea [26, 25] using *Google*, and OpenEphyra [46] using *Yahoo*. OpenEphrya also uses *Indri* [51], from the Lemur Project. *Indri* is a search engine developed at UMass that combines different features to allow more refined searches.

Nowadays, to our knowledge, there are only three available search engines able to answer computational requests, besides the aforementioned *Indri*: *Bing*, *Blekko* and *Entire Web*. This is due to the following facts: a) *Yahoo* API has been deprecated and is no longer available, as well as *Alta Vista*, which belongs to Yahoo!; b) *Google* API is deprecated as well, and the new one does not allow a significant number of queries free of charge; c) *AllTheWeb* also ceased to exist. In this way, the only survivors are these three. *Bing* is a traditional search engine, and works the same way as *Google* does. *Blekko*, on the other hand, has a limitation, as it only accepts queries with length of ten terms; any term besides the tenth is discarded blindly. However, it has an interesting feature: the so called *slashtags*. Slashtags allow to parameterize a query, by restricting the search domain (slashtags can restrict to a domain, such as *health* or *sport*). Finally, *Entire Web* only recently has an API.

2.6.2 (Semi-)Structured data sources

QA systems may also use semi-structured information sources. These are, usually, platforms where information is ensured to be correct, or at least assumed to be correct (for example, Wikipedia is an information source that usually is accurate but, as anyone can edit it, it may contain some errors). In this group we have some examples actually used by some systems, as explained next.

WordNet [30] is one of the most used sources. WordNet is used to search for synonyms, hyponyms, and other relations between words. This can be useful not only to query expansion, creating different queries meaning the same, but also when it comes to answer extraction and selection, as seen in Sections 2.1 and 2.3, as these relations may be important (for instance when clustering candidate answers or when matching terms from sentences with the question). DeepQA, Just.Ask and Prager et al. [38] use WordNet as explained.

DeepQA also uses other information sources, as Freebase [4], DBpedia [2] and Yago [52]. Freebase is a collection of facts about people, places and so on. DBpedia is a knowledge base that gathers Wikipedia semi-structured information into a structured platform, allowing easy access to facts and descriptions about anything found in Wikipedia. In fact, as pages use the same suffix, Just.Ask uses the Wikipedia page titles to index DBpedia pages and consult them to answer description questions. Yago is another online platform that consolidates information from various sources and tries to establish relations between entities.

Aranea uses yet other information sources, such as CIA World Factbook[2], 50states.com, Biography.com, and Internet Movie Database (IMDB)[3] to help answer questions about countries, people, and movies, respectively. As each information source is defined independently, Aranea uses a set of hand-crafted wrappers to access each one of these sources.

Wikipedia is also being used as a source more frequently, especially after 2007, when CLEF started to allow its usage. Systems like Joost [5], a Dutch QA system, and Just.Ask, as

[2]https://www.cia.gov/library/publications/the-world-factbook/index.html
[3]http://www.imdb.com/

referred before, use Wikipedia to answer not only description questions, but also some factoid questions like capitals, languages and other facts that can be easily extracted from Wikipedia pages.

2.7 Summary

In this chapter we started by presenting the state of the art in answer selection, followed by different works applied to WWBM and QA4MRE. From all presented solutions, except for those using patterns, only a few consider the context of the candidate answers where they appear, such as Lam et al. [22] (Section 2.2.2) and Echihabi et al. [11] (Section 2.1.3).

3 AnSelMo

Following some of the work presented in last chapter, we built a system, language independent as much as possible, that aims at answering multiple-answer questions. This chapter presents this system, AnSelMo (ANSwering SELection MOdule), starting with its architecture, followed by a dedicated section to each one of its modules: Pre-Processing, Counting, Lexical Approaches, Latent Semantic Analysis and Scoring.

3.1 Architecture

Figure 3.1 depicts the system's architecture. The core is the column of strategies that can be used to perform the answer selection task, with the grey box representing other techniques that may be developed and added to the system. Each of those modules will be described in greater detail in the following sections.

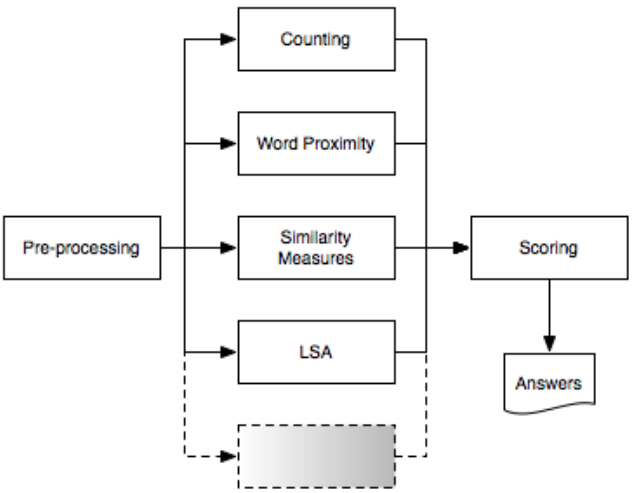

Figure 3.1: AnSelMo's Architecture.

The system starts by performing a pre-processing step (Section 3.2). From this step results the set of candidate answers[1] together with the snippets or documents that are related with them. These are then passed to one (or more) of the designed techniques. There are three classes of algorithms: Counting (Section 3.3), Lexical Approaches (Section 3.4) – which includes both Word Proximity and Similarity Measures – and Latent Semantic Analysis (Section 3.5). Finally, the scoring is done according to the weights given by each of the modules.

3.2 Pre-Processing

The pre-processing step includes different behaviors, depending on what is intended, which makes this step not totally language independent. In the following paragraphs we detail the possibilities of this step.

As sometimes there is the need to perform query formulation and IR steps (the cases where only the questions along with the candidate answers are given to the system), snippets need to be retrieved for each one of the hypothesis. This is done using two search engines, *Bing* and *Blekko* (see Section 2.6). For each question, distinct queries are envisaged, one for each hypothesis of answer. The process of query formulation involves different formats and filters (the system works with any combination of these):

- The answer can be quoted or not;
- The answer terms can appear before or after the question terms (AQ vs. QA format);
- Different filters can be applied (these are, actually, used throughout the whole system)[2]:
  - Wh-words–filter, where words such as *Where* and *Who* are eliminated;
  - Prepositions–filter, where prepositions like *At* are removed;
  - To Be–filter, that ignores different forms of the verb to be.

[1] Already known *apriori*.
[2] Henceforth, when referring to filtering in any of the strategies, any combination of these can be applied.

These filters are an example of language dependence of the system.

The system also works with local documents (or collection of documents). Each question should be bound to the document where the answer can (or should) be found. Thus, in this situation, the IR does not happen. We may also use Lucene to index those documents and later retrieve relevant snippets, given a query created by concatenating the question with the answer, filtered of stopwords, as before. Two indexation strategies are applied: in chunks of one or three sentences.

We may perform a simple form of anaphora resolution too. This only applies to documents that represent speeches and whose speaker is well defined. Pronouns like *I* and *me* are commonly used in such cases, so we substitute all those pronouns by the text's author.

3.3 Counting

This is one of the simplest modules and is based on the work of Lam et al. [22], described in Section 2.2.2. The authors performed experiments with this strategy, accomplishing interesting results. Recall that this strategy is based in the assumption that the correct candidate answer is the one that has the greatest number of hits when queering a search engine. The used queries are obtained as explained in Section 3.2, containing both the candidate answers and question terms.

3.4 Lexical Approaches

3.4.1 Word Proximity

The Word Proximity strategy is based on the assumption that answers occur close to question terms. The algorithm calculates the distance between each candidate answers' terms and the question terms in the neighborhood, that is, it weighs the distances, considering a maximum

radius[3], so that documents with too many references to an answer but not to the corresponding question terms are worth less. The algorithm was presented in Algorithm 1, in Section 2.2.2. This way, by consulting a set of documents, one can search for the answers and, in case of finding them, check within the radius for the question terms. Other variations were developed, in an attempt to minimize the linear decreasing given by the algorithm. The change was made in the line 8 of the algorithm, which calculates the scores given the distance between the answer and question terms. The different options available in our system are the following and their functions are depicted in Figure 3.2:

- Linear: $\frac{radius - |answerIdx - windowIdx|}{radius}$
- Quadratic: $\frac{radius^2 - |answerIdx - windowIdx|^2}{radius^2}$
- Cubic Root: $\frac{-\sqrt[3]{|answerIdx - windowIdx| - radius)}}{\sqrt[3]{radius}}$
- Cubic: $\frac{radius^3 - |answerIdx - windowIdx|^3}{radius^3}$
- Tetra (Fourth Power): $\frac{radius^4 - |answerIdx - windowIdx|^4}{radius^4}$

The point C in the figure represents the center point, that is, the point where an answer term was found. This is the $answerIdx$ in the function list (i in the original algorithm). Recall that, when an answer term is found, the algorithm looks for question terms in the surrounding terms, within a given radius; $windowIdx$ represents the index in the window and takes values between $-radius$ and $+radius$ (j in the original algorithm). The original function is the `Linear` one. All functions return a score between 0.0 and 1.0, representing respectively a greater or smaller distance to the center. The process is applied for each term present in the candidate answer, which will have a score of 0.0 if no question term is found within the radius.

The idea of the functions is, as noted, to diminish the impact of the linear decreasing of the original function. The `Quadratic`, `Cubic` and `Tetra` functions are all similar, only differing

[3]We use the term radius as it was introduced by Lam et al. [22], instead of window. Notice that the size of a window is twice the radius.

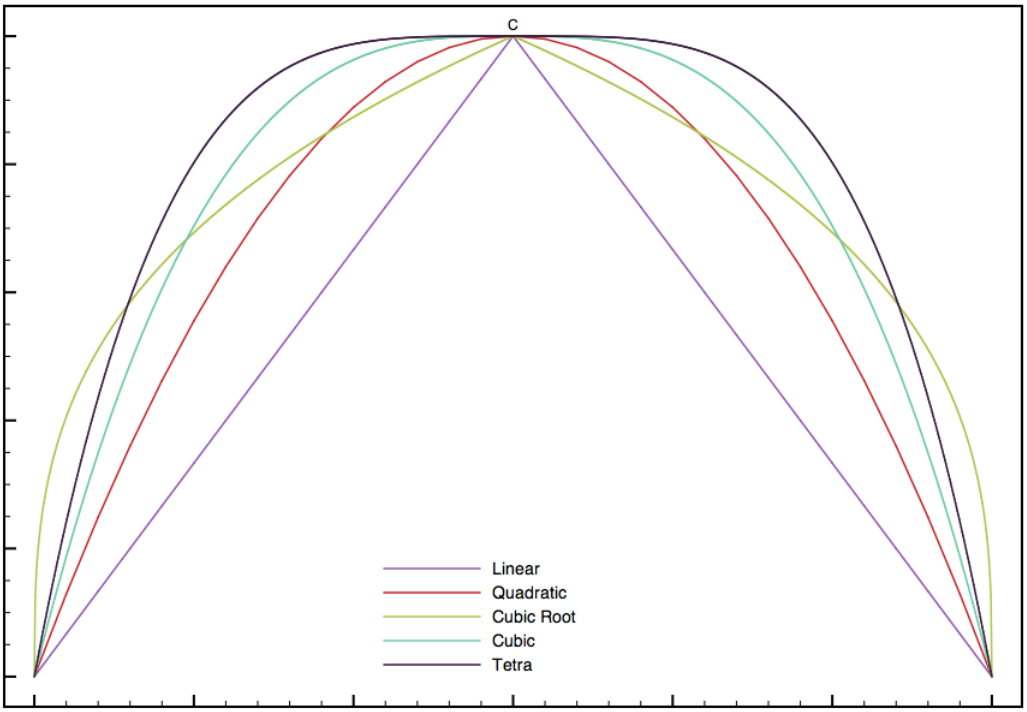

Figure 3.2: Score functions used in Word Proximity.

on how *late* they start to decrease the returned score. The `Cubic Root` function is an attempt to flatten the abrupt decreasing present in the powered functions (it is visible in the figure by its line, that crosses all others): it starts almost linearly and decreases faster only close to the extremes.

Consider the following example, from QA4MRE 2011 corpus: *Who is the founder of the SING campaign?* is the question, with two candidate answers, *Zackie Achmat* and *Annie Lennox* – the correct one. Given the passages *to have met Zackie Achmat, the founder of Treatment Action Campaign* and *And this is the name of Annie Lennox campaign, SING Campaign*[4], both in the associated document, the result for Word Proximity would be a score for each answer calculated as follows (assuming a radius of 10 and the `Linear` scoring algorithm):

- Zackie Achmat:

  Zackie distance to *founder* is 3, score is 0.7; distance to *campaign* is 7, score accumu-

[4]After anaphora resolution.

lates to $0.7 + 0.3$;

Achmat distance to *founder* is 2, score accumulates to $1.0 + 0.8$; distance to *campaign* is 6, score accumulates to $1.8 + 0.4$;

Total is $2.2/2 = 1.1$.

- Annie Lennox:

  Annie distance to *campaign* is 2, score is 0.8; distance to *SING* is 3, score accumulates to $0.8 + 0.7$; distance to *Campaign* is 4, score accumulates to $1.5 + 0.6$;

  Leenox distance to *campaign* is 1, score accumulates to $2.1 + 0.9$; distance to *SING* is 2, score accumulates to $3.0 + 0.8$; distance to *Campaign* is 3, score accumulates to $3.8 + 0.7$;

  Total is $4.5/2 = 2.25$.

We apply the algorithm to the top documents (passages) or snippets associated. However, each score corresponds to a single document, so a final score needs to be computed based on the calculated scores. For this we implemented three different scoring methods: a) `Mean`, which does the average of the scores; b) `Max`, which returns the maximum value; and c) `sMean`, which ignores the maximum and minimum value (considering them outliers) and computes the average for the remaining values.

3.4.2 Similarity Measures

Other approach used by our system is based on similarity measures. For this we use the notion of extents, that is, a passage that includes each term of a given query at least once (as presented by Clarke et al. [9] – see Section 2.2.1). The used queries are simply the questions and answers, seen as bags of words, and filtered of stopwords. Thus, we will have an extent for the question (question extent) and many extents as answers (answer extents). As before, those can also be filtered.

This is the first definition we used to create our strategy, and is henceforth referred simply as `Extents`. However, as an extent must contain all query terms, it may happen that no extent is created because not all terms exist in the text. Thus, if we are not able to identify all query terms in the text, then an extent with only the existing terms will be created. Also, if a given extent has length below a given threshold, we expand it by n terms at the start and end of the extent; otherwise, the extent would be too short to use the similarity measures. On the other hand, the extents may end up being too large, due to the requirement of meeting such a restrictive constraint. These will damage the similarity measures, because the extents would be too similar.

With this in mind, we developed another strategy to create our extents. This is based on POS tagging[5]. The idea is to have important words (read nouns and verbs present in the query) to contribute with some weight to the extent. The extent has a score threshold, which, if surpassed, defines the extent. Table 3.1 shows the scores attributed to each POS. The threshold is defined by two parameters: the tag threshold and the *others* threshold. The later is set, empirically, to 8.0, while the former is defined in function of the query, and is set to half the total present in the query. This way we can create extents that contain only parts of the query (thus, reducing their size), but that are still large enough to apply the similarity measures (for example, if the tag threshold is set to 12.0, and we find three Proper Nouns together (3 times 4.0), we still need to find other eight words (8 times 1.0) to complete the extent[6]). This strategy is called `Extents Points`.

POS Tag	**Score**
Proper Noun	4.0
Common Noun	2.5
Verb	1.5
Others	1.0

Table 3.1: Scores for each POS tag.

The extents, from any of the two strategies, are then compared against each other (question extent versus each one of the answer extents). The most widely used similarity measures that

[5]Another language dependence point of the system.
[6]Whenever necessary, this expansion is done evenly for both sides of the extent.

do not penalize word order are used in our system: `Overlap`, `Jaccard` and `Dice`, as defined in Equations 3.1 to 3.3.

$$\text{Overlap}(X, Y) = \frac{|X \cap Y|}{\min(|X|, |Y|)} \tag{3.1}$$

$$\text{Jaccard}(X, Y) = \frac{|X \cap Y|}{|X \cup Y|} \tag{3.2}$$

$$\text{Dice}(X, Y) = 2 \times \frac{|X \cap Y|}{|X| + |Y|} \tag{3.3}$$

We made some modifications to the interpretation given to intersection and union. Usually these similarity measures are used with sets, that is, identical words will collapse into one unique word, either in intersection or union. However, an extent containing only once a given term hardly represents the same passage as another extent containing five times that term. We wanted, thus, to account that. Respecting the similarity measures properties, such as returning values between 0.0 and 1.0, we defined intersection and union in such way that can be applied to these similarity measures:

Intersection The intersection between two extents (two bags of words) is the bag of words that contain all words that co-occur in both extents for each time they co-occur;

Union The union between two extents (two bags of words) is a bag of words containing all words that do not co-occur in both extents plus the words present in the intersection.

An example illustrates better these concepts. Given two extents $E_1 = \{A, B, C, B\}$ and $E_2 = \{A, A, B, B\}$, the intersection will be the bag of words $I = \{A, B, B\}$ and the union the bag of words $U = \{A^*, B^*, B^{**}, C, A\}$, where A^*, B^* and B^{**} are the words from the intersection. As the similarity measures only care about sizes, the union size can easily be computed as $|U| = |E_1| + |E_2| - |I|$.

To illustrate how Similarity Measures work (with `Jaccard`, for instance), we take as example

the one provided for Word Proximity: *Who is the founder of the SING campaign?* is the question, with two candidate answers, *Zackie Achmat* and *Annie Lennox* – the correct one. Three extents are defined: the question extent, and two answers extents. The corresponding bag of words could be, respectively, {*founder, SING, campaing*}, {*Zackie, Achmat*} and {*Annie, Lennox*}, and would appear in the final extents. The following extents (as example) would produce the scores below:

Question extent {*of, Annie, Lennox, campaign, SING, Campaign, In, November, of*};

Zackie Achmat extent {*to, have, met, Zackie, Achmat, the, founder, of, Treatment, Action, Campaign*}; Two terms are common: {*of*, *Campaign*}. Score = $2/(9 + 11 - 2) = 0.11$;

Annie Lennox extent {*And, this, is, the, name, of, Annie, Lennox, campaign, SING, Campaign*}; Six terms are common: {*of*, *Annie*, *Lennox*, *campaign*, *SING*, *Campaign*}. Score = $6/(9 + 11 - 6) = 0.43$.

3.5 Latent Semantic Analysis

The last technique used by our system is based in LSA. As described in Section 2.5, LSA computes a matrix A and then applies SVD to it. After SVD application, both the question and answers are projected over the space defined by matrix V^T. This will result in a set of vectors representing the question and each answer under the space V^T. There are then two approaches that can be used:

A1 In the first approach, we compare the question with each passage present in matrix V^T, selecting the most similar passage to the question. Then we select as correct answer the most similar answer to the selected passage.

A2 In the other approach, we directly compare the question to each answer in the space defined by matrix V^T, selecting as correct answer the one most similar to the question.

Basically, the difference between the two is the indirection point added in the first approach, where an additional comparison is made (the most similar sentence to the question and answer).

To compute similarity in the space defined by matrix V^T, we use the cosine of the angle defined by the vectors representing the passages under comparison ($\mathbf{x}$ and $\mathbf{y}$):

$$sim_{\cos}(\mathbf{x}, \mathbf{y}) = \frac{\mathbf{x} \cdot \mathbf{y}}{\|\mathbf{x}\|\|\mathbf{y}\|} = \frac{\sum_{i=1}^{n} x_i y_i}{\sqrt{\sum_{i=1}^{n} x_i^2}\sqrt{\sum_{i=1}^{n} y_i^2}} \tag{3.4}$$

In the definition of the semantic space, we explore different weighting strategies. According to Sahlgren [43], the most effective weighting schemes for small contexts are based on the simple frequency (or even binary), that do not use global weights, such as raw or dampened counts. We explore raw and normalized frequency. As normalization factors we use both the max coordinate value and the sum of the values of the coordinates.

3.6 Scoring

The scoring step is the final one, and receives the scores from each technique. It can combine those, according to given weights, and then sorts the candidate answers decreasingly. The top-most answer is chosen, except in cases where the question is negative, that is, questions that contain the word *not*. For example, for the question *Which of these plays is not written by Shakespeare?*, probably the candidate answer with the least score is the one less related with the question and, thus, the correct answer. In these cases, we choose as correct answer the one with the least score. If no answer has a score, then the system will not give an answer.

3.7 Summary

This chapter described AnSelMo, our ANSwering SELection MOdule. The system is language independent as much as possible and has, actually, four selection modules: Counting, Word

Proximity, Similarity Measures and Latent Semantic Analysis. These are used to select the correct candidate answer to a question, among the different candidate answers available.

4 Evaluation

This chapter presents the application of AnSelMo to each scenario and respective evaluation. We first describe the experimental setup in Section 4.1, followed by the attained results for each technique. Throughout those sections, best results will be presented, always referring to the remaining results in annex. We end in Section 4.6 with a discussion of the results.

4.1 Experimental Setup

First we detail the data set used in our experiments and, then, the evaluation metrics proposed to evaluate the system.

4.1.1 Data Sets

In our evaluations we used different data sets, depending on the task being evaluated. As previously stated, we considered three scenarios: WWBM and QA4MRE, which are tasks with multiple-answer questions where one hypothesis is the correct one, and Just.Ask, using as candidate answers the hypothesis extracted by the original system.

4.1.1.1 'Who Wants to Be Millionaire?'

Regarding WWBM, we used two corpora in two different languages: Portuguese and English. For the latter we collected a set of questions from some editions of the computer/console game 'Who Wants to Be a Millionaire?'[1]. We extracted a subset of 100 questions, randomly

[1]The question set is available at *GameFAQs*, at http://www.gamefaqs.com/gba/582399-who-wants-to-be-a-millionaire/faqs/37922

picked, without caring about difficulty (in the human point of view the initial ones are the easiest). For the Portuguese set, we used a corpus of 100 questions from the local broadcast of the WWBM TV show, manually transcribed, also randomly chosen among a total of 180 questions.

The description of both corpora can be seen in Table 4.1.

	English		**Portuguese**	
	Questions	**Answers**	**Questions**	**Answers**
Number	100	400	100	400
Total words	981	591	1151	752
Unique words	376	406	558	593
Longest	17	5	18	6
Shortest	3	1	3	1
Average	9.81	1.47	11.51	1.88

Table 4.1: WWBM corpora characteristics.

4.1.1.2 Question Answering for Machine Reading Evaluation

Considering the QA4MRE challenge, the English 2011 corpus was used. It comprises three topics: "Aids", "Climate Change" and "Music and Society". The characteristics of the corpus can be consulted in Table 4.2.

	Questions	**Answers**
Number	120	600
Total words	1256	2338
Unique words	508	950
Longest	21	15
Shortest	4	1
Average	10.47	3.90

Table 4.2: QA4MRE corpus characteristics.

4.1.1.3 Just.Ask

Finally, for Just.Ask experiments we used a set of 558 questions. Those questions are a subset of the original corpus used by Just.Ask, and correspond to those for which it is able to extract

the correct answer and place it in the top 50 candidates. Thus, at least one or more candidate answers are the correct answer in these questions, having each question at most 50 candidates. The corpus characteristics are in Table 4.3.

	Questions	**Answers**	
Number	558	13575	**Candidates by Question**
Total words	4143	22559	
Unique words	1274	7634	
Longest	19	12	50
Shortest	3	1	1
Average	7.42	1.66	24.33

Table 4.3: Just.Ask corpus characteristics.

4.1.1.4 Discussion

Table 4.4 resumes the different corpora characteristics (the averages by questions and answers). Regarding the WWBM corpus, we can see that Portuguese is a much more rich and verbose language, using about 200 new words and in average two more words per question.

	Questions		**Answers**	
Corpus	**Total words**	**Unique words**	**Total words**	**Unique words**
WWBM (EN)	9.81	3.76	1.47	1.01
WWBM (PT)	11.51	5.58	1.88	1.48
QA4MRE	10.47	4.23	3.90	1.58
Just.Ask	7.42	1.28	1.66	0.56

Table 4.4: Comparison of the different corpora average statistics.

We should also note that we opted not to use a translation of the same corpus because some questions only make sense in their original language, such as questions about TV shows or important people of the corresponding country. For example, the Portuguese question *O que é uma pescadinha de rabo na boca?* refers to a traditional Portuguese dish. Translated, it would be something like *What is a whitefish of tail in the mouth?*, which does not relate with any dish. Thus, although with different questions, we think it makes more sense to compare the behavior of the system with questions from the original language.

Another important point is the difference between WWBM and QA4MRE corpora. If we measure the number of words and unique words by question, we see that the numbers for the QA4MRE corpus are greater than for the English WWBM corpus. This may have to do with the nature of the challenge: QA4MRE is a Machine Reading task and may have more complex questions, requiring knowledge of more words and meanings. On the other hand, these values are smaller than those for the Portuguese WWBM corpus, which corroborates the initial suspicion. On the other hand, QA4MRE answers tend to be longer than the others. Once again, this is strictly related with the task nature.

Finally, we can see that Just.Ask corpus is quite similar to the WWBM English corpus, except for the unique words value, which is smaller. However, this value is probably diluted by Just.Ask corpus being a much bigger corpus.

4.1.2 Evaluation Measures

We evaluate our system according to different metrics. Those aim at better understanding how the system's techniques perform, being also able to compare the system's performance in each scenario, although not directly.

QA4MRE task has its own evaluation measure [34]. The measure, called c@1, rewards a system that opts by not answering a question, instead of answering it wrongly. The reward is in function of the number of correct answers. c@1 is defined as follows:

$$c@1 = \frac{1}{N}(n_R + n_U \frac{n_R}{N}),$$

where N is the number of questions, n_R the number of correct answers and n_U the number of unanswered questions. This metric is specially useful to compare the system against others participating in the QA4MRE task. Moreover, it will be used in all scenarios.

It may be also interesting to evaluate the system only considering the number of correct answers. Thus, accuracy will also be contemplated[2]:

[2]Notice that c@1, when answering to all questions, boils down to accuracy.

$$Accuracy = \frac{n_R}{N}.$$

A third evaluation measure to be applied is Mean Reciprocal Rank (MRR). MRR weighs the answers given to each question according to their rank among other candidates to the same question and can be useful to measure how far, overall, the correct answers are from the top:

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

with $rank_i$ being the rank of the correct answer to the i-th question, among the answers returned. We look to all candidate answers, so unless the system does not answer the question (in this case the MRR will be 0.0 for that question), the correct candidate answer will always contribute with some score for MRR – for instance, in WWBM the lowest nonzero score would be 1/4.

4.2 Counting

The Counting technique was only applied to WWBM, as only in this scenario an IR step is needed. In QA4MRE the answer should be found in the given document and with Just.Ask the candidate answers are already extracted from a search engine. Because the built system allows different parameterizations (for example, regarding query formulation[3], we can have queries created in eight different forms – the combination of the filters, AQ or QA format and the usage of quotes), the number of different runs grows exponentially, specially later, when using more complex strategies. As the focus is answer selection, and not query formulation, we want to evaluate how well our techniques perform and this is only possible by comparing them in the same scenarios, that is, using the same information. Thus, we also used the Counting technique to narrow our testing experiments to only a few (that is, using a set of selected parameterizations for the query formulation step). The best results (with accuracies

[3]Note that only WWBM experiments require a query formulation step.

above 60%, for English[4]) were then used to run some experiments with Word Proximity, now only with the English corpus. Those runs were also used to understand which are the best parameterizations to use (that is, the ones that apparently provide the best snippets). Note that the best parameterizations may work better with some techniques and the worst with others. However we believe that the assumption holds and that the best parameterizations can, in fact, generate the best results overall.

Tables 4.5 and 4.6 show the best results with Counting technique. The columns AQ, Quote and Filter respect to the passage retrieval step (in the query creation): for AQ mode (answer concatenated with the question), the column has *true* as value, and for QA (question concatenated with the answer), *false*. Quote represents the quotation of the answer in the query and the filters initials represent the different possible filters presented in Section 3.2.

Search	AQ	Quote	Filter	Acc.	MRR	C@1
BING	true	true	WH_PROP_BE	0.70	0.802	0.714
BING	true	true	BE	0.67	0.783	0.683
BING	true	true	NO	0.66	0.783	0.673
BING	false	true	NO	0.73	0.834	0.745
BING	false	true	BE	0.69	0.812	0.704
BING	false	true	PROP_BE	0.69	0.807	0.704
BLEKKO	true	true	WH_PROP_BE	0.67	0.802	0.723
BLEKKO	true	true	WH_PROP	0.66	0.794	0.718
BLEKKO	true	true	WH	0.65	0.787	0.680
BLEKKO	true	false	PROP_BE	0.62	0.767	0.650
BLEKKO	true	false	PROP	0.60	0.767	0.624
BLEKKO	true	false	BE	0.58	0.748	0.604

Table 4.5: Best results accomplished with Counting, for WWBM English corpus. The whole table may be consulted in annex (Tables A.1 and A.2).

4.3 Word Proximity

This section reports the results for Word Proximity, which are divided for each scenario were it was applied.

[4]For Portuguese, the identical runs were selected.

Search	AQ	Quote	Filter	Acc.	MRR	C@1
BING	true	true	WH_PROP_BE	0.50	0.679	0.488
BING	true	true	WH_BE	0.50	0.677	0.486
BING	true	true	PROP_BE	0.50	0.675	0.488
BING	false	true	WH_BE	0.49	0.678	0.475
BING	false	true	BE	0.49	0.673	0.475
BING	false	true	WH_PROP	0.49	0.670	0.477
BLEKKO	true	true	WH_PROP_BE	0.31	0.565	0.237
BLEKKO	true	true	WH_PROP	0.29	0.555	0.220
BLEKKO	true	true	PROP_BE	0.29	0.554	0.203
BLEKKO	true	false	WH_PROP_BE	0.30	0.558	0.228
BLEKKO	true	false	NO	0.30	0.556	0.277
BLEKKO	true	false	BE	0.30	0.553	0.276

Table 4.6: Best results accomplished with Counting, for WWBM Portuguese corpus. The whole table may be consulted in annex (Tables A.3 and A.4).

4.3.1 'Who Wants to Be Millionaire?'

For the WWBM data set, we continued the experiments aiming the reduction of the test set. This was done with the English corpus, for the parameterizations presented in the last section. We ran tests with different values for radius and number of passages (respectively 20, 40, 60 and 5, 10, 20). This resulted in 216 different tests, which can be consulted in annex (Table B.1). In Table 4.7 are the two best results for both search engines.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	false	true	PROP_BE	0.65	0.772	0.663	20	20
BING	true	true	BE	0.65	0.771	0.663	60	20
BLEKKO	true	false	WH_PROP_BE	0.68	0.806	0.730	20	20
BLEKKO	true	false	WH_PROP_BE	0.68	0.805	0.720	20	5

Table 4.7: Best results accomplished with Word Proximity, for WWBM English corpus, using `Mean` as combination method and `Linear` as scoring algorithm. The whole table is in annex (Table B.1).

Although it is not visible in this table, the top 6 results for *Bing* and *Blekko* comprise, respectively, six and five runs with a number of passages equal to 20. This shows that the strategy benefits from analyzing more documents. For this reason we ended up choosing two parameterizations for each search engine to test more deeply the possibilities of Word

Proximity technique. The selected parameterizations are shown in Table 4.8.

Search	AQ	Quote	Filter	Passages
BING	false	true	PROP_BE	20
BING	true	true	BE	20
BLEKKO	true	false	WH_PROP_BE	20
BLEKKO	true	true	WH_PROP_BE	20

Table 4.8: Parameterizations used in further tests to WWBM scenario.

These parameterizations were used to test different values for radius. The first tests only used three values (20, 40 and 60), so we ran more tests, now with values for radius of 3, 4, 5, 6, 10, 30, 50, 70 and 80. Table 4.9 shows the best attained results.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	true	true	BE	0.66	0.781	0.673	70	20
BING	true	true	BE	0.65	0.774	0.663	80	20
BING	false	true	PROP_BE	0.65	0.772	0.663	30	20
BING	false	true	PROP_BE	0.64	0.772	0.653	80	20
BING	false	true	PROP_BE	0.64	0.770	0.653	70	20
BLEKKO	true	true	WH_PROP_BE	0.77	0.850	0.824	5	20
BLEKKO	true	true	WH_PROP_BE	0.75	0.842	0.830	4	20
BLEKKO	true	true	WH_PROP_BE	0.72	0.833	0.809	6	20
BLEKKO	true	false	WH_PROP_BE	0.72	0.823	0.773	30	20
BLEKKO	true	true	WH_PROP_BE	0.71	0.827	0.777	10	20

Table 4.9: Best results accomplished with Word Proximity, for WWBM English corpus, using `Mean` as combination method and `Linear` as scoring algorithm, for different radius values. The whole table is in annex (Table B.2).

Regarding the different algorithms and combination methods, we choose the two best radius values (5 and 70), and used them with the best parameterization for *Bing* and *Blekko*, according with Table 4.9. The test set for scoring algorithms has `Mean` as fixed combination method, and the combination methods test set has `Linear` fixed as scoring algorithm. The test containing both `Mean` as combination method and `Linear` as scoring algorithm is common to both test sets and it corresponds to the darker row on Table 4.10.

Tests for Portuguese corpus used the same parameterizations as English tests (Table 4.8), for radius values of 5, 20 and 70. Table 4.11 shows the accomplished results. It is clear that *Blekko*

	BING, AQ, Quote, BE	Rad. 5			Rad. 70		
		Acc.	**MRR**	**C@1**	**Acc.**	**MRR**	**C@1**
Scoring Algorithm	Quadratic	0.61	0.756	0.622	0.66	0.785	0.673
	Cubic	0.61	0.756	0.622	0.68	0.795	0.694
	Tetra	0.63	0.766	0.643	0.68	0.795	0.694
	Cubic_Root	0.64	0.771	0.653	0.68	0.795	0.694
Combination Method	Linear/Mean	0.62	0.766	0.632	0.66	0.781	0.673
	S_Mean	0.63	0.768	0.649	0.68	0.792	0.694
	Max	0.51	0.707	0.520	0.59	0.759	0.602
Scoring Algorithm	Quadratic	0.77	0.852	0.824	0.71	0.769	0.809
	Cubic	0.77	0.852	0.824	0.74	0.784	0.844
	Tetra	0.77	0.852	0.824	0.74	0.784	0.844
	Cubic_Root	0.77	0.852	0.824	0.72	0.774	0.821
Combination Method	Linear/Mean	0.77	0.850	0.824	0.69	0.759	0.787
	S_Mean	0.75	0.838	0.816	0.67	0.748	0.764
	Max	0.68	0.806	0.719	0.66	0.739	0.752
	BLEKKO, AQ, Quote, WH_PROP_BE	**Acc.**	**MRR**	**C@1**	**Acc.**	**MRR**	**C@1**
		Rad. 5			**Rad. 70**		

Table 4.10: Results with Word Proximity, for WWBM English corpus, using different combination methods and scoring algorithms.

is not able to achieve the same results for Portuguese; for this reason, the tests for scoring algorithms and combination methods were only applied to the best *Bing* parameterization (Table 4.12).

4.3.2 Question Answering for Machine Reading Evaluation

For QA4MRE, two strategies were adopted. The first only uses the documents to apply the Word Proximity algorithm. The other uses Lucene to retrieve relevant snippets, given a query (created the same way it was done for search engines presented before).

Results are present in Table 4.13, where other runs with Lucene were excluded due to their low results (both shown runs use the indexation in chunks of three sentences).

To test the impact the different scoring algorithms, we ran the tests for the two best radius values: 20 and 50. Results are shown in Table 4.14.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	true	true	BE	0.51	0.653	0.561	70	20
BING	false	true	PROP_BE	0.50	0.636	0.565	70	20
BING	true	true	BE	0.49	0.641	0.549	20	20
BING	false	true	PROP_BE	0.48	0.63	0.557	5	20
BING	false	true	PROP_BE	0.48	0.63	0.552	20	20
BING	true	true	BE	0.45	0.603	0.518	5	20
BLEKKO	true	false	WH_PROP_BE	0.16	0.207	0.272	70	20
BLEKKO	true	false	WH_PROP_BE	0.15	0.198	0.257	20	20
BLEKKO	true	false	WH_PROP_BE	0.15	0.194	0.230	5	20
BLEKKO	true	true	WH_PROP_BE	0.14	0.173	0.247	70	20
BLEKKO	true	true	WH_PROP_BE	0.13	0.163	0.230	20	20
BLEKKO	true	true	WH_PROP_BE	0.12	0.153	0.215	5	20

Table 4.11: Best results accomplished with Word Proximity, for WWBM Portuguese corpus, using `Mean` as combination method and `Linear` as scoring algorithm, for different radius values.

	BING, AQ, Quote, BE	Rad. 5			Rad. 70		
		Acc.	MRR	C@1	Acc.	MRR	C@1
Scoring Algorithm	Quadratic	0.47	0.617	0.541	0.50	0.648	0.550
	Cubic	0.47	0.618	0.541	0.50	0.646	0.550
	Tetra	0.47	0.618	0.541	0.50	0.649	0.550
	Cubic_Root	0.47	0.618	0.541	0.51	0.654	0.561
Combination Method	Linear/Mean	0.45	0.603	0.518	0.51	0.653	0.561
	s_Mean	0.44	0.598	0.510	0.53	0.662	0.583
	Max	0.43	0.597	0.495	0.50	0.645	0.550

Table 4.12: Results with Word Proximity, for WWBM Portuguese corpus, using different combination methods and scoring algorithms.

	Document								Luc. BC	Luc.
Rad.	50	20	30	70	40	60	10	5	50	5
Acc.	0.317	0.317	0.308	0.300	0.300	0.292	0.242	0.233	0.075	0.058
MRR	0.523	0.518	0.521	0.517	0.515	0.514	0.452	0.391	0.145	0.103
C@1	0.343	0.351	0.334	0.323	0.325	0.314	0.286	0.303	0.129	0.105

Table 4.13: Best results accomplished with Word Proximity, for QA4MRE 2011 corpus, using `Linear` as scoring algorithm, for different radius values. Last columns contain runs using Lucene, indexed with and without the Background Collection.

4.3.3 Just.Ask

We performed two different experiments with Just.Ask: using AnSelMo as a full system and using it as an answering selection module, together with Just.Ask. In this case, the scores

	Linear		Quadratic		Cubic		Tetra		Cubic_Root	
Rad.	20	50	20	50	20	50	20	50	20	50
Acc.	0.317	0.317	0.317	0.292	0.317	0.275	0.300	0.275	0.300	0.267
MRR	0.518	0.523	0.515	0.507	0.514	0.497	0.506	0.498	0.506	0.493
C@1	0.351	0.343	0.351	0.316	0.351	0.298	0.333	0.298	0.333	0.289

Table 4.14: Results accomplished with Word Proximity, for QA4MRE 2011 corpus, using different scoring algorithms, for 20 and 50 as radius values.

returned by the module will be the initial scores of each candidate answer. Table 4.15 shows the results for the first approach, and Table 4.16 the results for the other[5]. Baseline is the best result attained by Just.Ask.

Algorithm	**Radius**	**Accuracy**	**MRR**	**Top-3 Accuracy**
Baseline		0.599	0.714	0.803
Linear	10	0.097	0.254	0.270
Cubic	10	0.089	0.247	0.258
Cubic_Root	10	0.088	0.246	0.265
Cubic	40	0.081	0.240	0.267
Cubic_Root	40	0.079	0.239	0.256
Cubic	30	0.077	0.239	0.267
Linear	30	0.075	0.242	0.277
Cubic_Root	30	0.075	0.236	0.250
Linear	40	0.072	0.236	0.259

Table 4.15: Best results accomplished with Word Proximity, for Just.Ask corpus, using AnSelMo as a full system.

4.3.4 Discussion

This section reported different results for Word Proximity and its variations. The first thing we can understand is that the radius value is not trivial to get and depends a lot on the source information: the difference between *Bing* and *Blekko* documents is noticeable in their optimal radius values (from 70 to only 5, respectively). As we already mentioned before (Section 2.6), *Blekko* is a *spam-free* engine. This implies less documents but, at the same time, documents with greater quality, which can explain the non-need of greater values for radius. Parsing

[5]Just.Ask calculates Top-3 accuracy instead of c@1.

Algorithm	Radius	Accuracy	MRR	Top-3 Accuracy
Cubic	40	0.612	0.723	0.805
Cubic_Root	40	0.610	0.721	0.803
Cubic	30	0.608	0.721	0.805
Cubic_Root	30	0.608	0.720	0.803
Linear	30	0.608	0.721	0.801
Linear	40	0.606	0.719	0.805
Baseline		0.599	0.714	0.803
Cubic	10	0.597	0.711	0.792
Cubic_Root	10	0.596	0.711	0.794
Linear	10	0.589	0.705	0.787

Table 4.16: Best results accomplished with Word Proximity, for Just.Ask corpus, using AnSelMo as an answering selection module.

problems could also explain that difference, as *Bing* documents may be left with more noise, thus requiring greater values for radius.

Regarding the scoring algorithms, and having `Linear` as baseline, two standout: `Cubic Root` and `Tetra`. Although not much conclusions can be drawn from the results, both surpass `Linear`. Their flattened curves improve the scoring algorithm, turning the technique less prone to choosing a bad parameterization – for example, a not appropriate radius value –, having greater impact in potential worst results.

For combination methods, `Mean` works better and `sMean` seems to not have a great impact (although it can improve results for some parameterizations). `Max`, on the other hand, penalizes the system, probably because it cannot take advantage of analyzing more documents.

For Portuguese corpus, using *Blekko*, results are really poor. In fact, for almost all questions we got no hits. Again, *Blekko* does not index all web, having instead a set of 'safe' domains. Thus, we believe that there are few Portuguese sites indexed, resulting in poor quality documents (or their nonexistence).

In QA4MRE, we can see that a value for radius of 20 is better, although there is no significant impact in using greater values. However, smaller values have a negative impact on the technique performance for this task. Regarding the different scoring algorithms, we can see

that they tend to damage the performance for greater values of radius, when compared with `Linear`.

For Just.Ask, the results are quite different, depending on the approach used. When using AnSelMo as a full system, results show one of its limitations: the techniques perform worst by using only snippets instead of documents. Also, candidates may be related and redundancy is not taken into consideration. However, using it together with Just.Ask allows the QA system to improve its results (best run answers correctly to 8 more questions). In the last approach, we can see that the greater the radius, the better. In fact, with radius equal to 10 Just.Ask performs below the baseline. Note that a radius of 40 is enough to scan the entire snippets, and for this reason we did not increase that value. It is also possible to improve Top-3 accuracy and MRR in exchange of some correct answers. This may be important to some systems that perform further steps or simply return the top-n results.

4.4 Similarity Measures

As with Word Proximity, results are divided for each scenario where the Similarity Measures technique was applied.

4.4.1 'Who Wants to Be Millionaire?'

For WWBM, tests were run using the same parameterizations of Table 4.8. Both `Extents` and `Extents Points` were used to WWBM English and Portuguese corpora.

The best results for WWBM English corpus are shown in Tables 4.17 and 4.18.

The best results for WWBM Portuguese corpus are shown in Tables 4.19 and 4.20. *Blekko* was not considered for these experiments.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.53	0.705	0.541	Dice	10
BING	true	true	BE	0.52	0.704	0.530	Dice	20
BING	false	true	PROP_BE	0.52	0.703	0.530	Jaccard	10
BLEKKO	true	true	WH_PROP_BE	0.67	0.753	0.764	Dice	20
BLEKKO	true	true	WH_PROP_BE	0.67	0.752	0.764	Jaccard	20
BLEKKO	true	true	WH_PROP_BE	0.61	0.722	0.695	Dice	5

Table 4.17: Best results accomplished with Similarity Measures, for WWBM English corpus, using `Extents` and different similarity measures. The whole table may be consulted in annex (Table C.1).

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.72	0.817	0.763	Dice	20
BING	true	true	BE	0.68	0.791	0.721	Dice	20
BING	false	true	PROP_BE	0.66	0.787	0.700	Jaccard	20
BLEKKO	true	false	WH_PROP_BE	0.60	0.706	0.690	Dice	20
BLEKKO	true	false	WH_PROP_BE	0.59	0.696	0.679	Dice	10
BLEKKO	true	true	WH_PROP_BE	0.59	0.688	0.696	Dice	20

Table 4.18: Best results accomplished with Similarity Measures, for WWBM English corpus, using `Extents Points` and different similarity measures. The whole table may be consulted in annex (Table C.2).

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.40	0.575	0.448	Jaccard	5
BING	true	true	BE	0.39	0.579	0.425	Dice	10
BING	true	true	BE	0.39	0.579	0.425	Jaccard	10

Table 4.19: Best results accomplished with Similarity Measures, for WWBM Portuguese corpus, using `Extents` and different similarity measures. The whole table may be consulted in annex (Table C.3).

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.44	0.577	0.519	Dice	20
BING	false	true	PROP_BE	0.44	0.577	0.519	Jaccard	20
BING	true	true	BE	0.43	0.578	0.495	Dice	5

Table 4.20: Best results accomplished with Similarity Measures, for WWBM Portuguese corpus, using `Extents` and different similarity measures. The whole table may be consulted in annex (Table C.4).

4.4.2 Question Answering for Machine Reading Evaluation

Once again, two approaches were taken: analyzing the document or using passages retrieved from Lucene. However, results using Lucene are as low as before, reason why we do not present them. In Table 4.21 we present the results for both `Extents` and `Extents Points` techniques.

	Extents			**Extents Points**		
Sim.	**Overlap**	**Jaccard**	**Dice**	**Overlap**	**Jaccard**	**Dice**
Acc.	0.242	0.267	0.267	0.192	0.233	0.233
MRR	0.502	0.532	0.532	0.386	0.402	0.402
C@1	0.248	0.273	0.273	0.248	0.301	0.301

Table 4.21: Results accomplished with Similarity Measures, for QA4MRE 2011 corpus, using different similarity measures and both `Extents` and `Extents Points`.

4.4.3 Discussion

In this section we reported results obtained with our technique of Similarity Measures, using extents. Two strategies were applied to extents' creation: one is based in the original definition and the other in POS tagging (see Section 3.4.2).

The first thing that comes out is the difference between performances of `Overlap` against `Dice` or `Jaccard`. One explanation for this relies in one known Overlap limitation: if one of the extents is contained by the other, then the ratio between the intersection and the number of elements of the minimal extent will be 1.0. Sometimes this will end with an incorrect answer having the top score or two or more answers be tied for the top score. Also, a deeper analysis to the results showed that the correct answer is often only approximately 0.1 or less from the top. This can be proven by the high MRR results that `Overlap` runs have, despite the lower accuracies accomplished (see Annex D).

Regarding the two strategies, it is interesting to see that one works better with *Bing* than with *Blekko* and vice-versa. However, although that for *Blekko* the results are a bit lower with `Extents Points`, the increase in runs with *Bing* is significant. Also, it seems that, again, the

techniques perform better when analyzing more documents.

Another look to the results showed that the ordination for the candidate answers between the runs from `Extents` and `Extents Points` is fairly different. For example, for a given question, one answers correctly and the other has that answer as last in its order, but for another question the first run cannot give a score to the correct answer while the other can place it in the top. This makes us believe that each approach may be adequate for some questions types and not others, probably depending on the question and answers sizes, terms, etc. Thus, a mixture of both runs may have a significant impact on the system's performance.

For QA4MRE, it is interesting to note that the second approach (`Extents Points`) damages slightly the accuracy but increases that much the c@1 measure.

4.5 Latent Semantic Analysis

In this section we report the results attained by using LSA as explained in Section 3.5. The scenarios to which LSA was applied are WWBM and QA4MRE, detailed in the following subsections. Matrix A is created with *terms by documents* when applied to WWBM and *terms by passages* for QA4MRE[6]. The reason behind this option is that in QA4MRE scenario we only have one document for all candidate answers, whereas in WWBM we have a set of documents for each candidate answer. Thus, creating the matrix A with *terms by sentences*, in QA4MRE, will allow a greater segmentation of the data, which is not necessary in WWBM and that would be too costly, computationally – documents are larger and in greater quantity.

A1 and A2 are the approaches described in Section 3.5 and `Freq.`, `Norm. (Sum)`, and `Norm. (Max)` are the different weighting strategies explored.

[6]In Just.Ask scenario we only have snippets and, thus, this technique cannot be applied.

4.5.1 'Who Wants to Be Millionaire?'

For WWBM we opted to use the apparent best formulations: *Bing*, AQ, Quoting and BE as filter and *Blekko*, AQ, Quoting and WH_PROP_BE as filter. Experiments used a different number of passages (1, 5 or 10), meaning that matrix A will have at most four, twenty or forty documents, respectively. Table 4.22 shows the results accomplished[7].

Search	AQ	Quote	Filter	Acc.	C@1	App.	Passages	Norm.
BING	true	true	BE	0.45	0.504	A2	10	Max
BING	true	true	BE	0.40	0.444	A2	5	Max
BING	true	true	BE	0.37	0.444	A2	1	-
BLEKKO	true	true	WH_PROP_BE	0.28	0.381	A2	10	-
BLEKKO	true	true	WH_PROP_BE	0.26	0.377	A1	5	-
BLEKKO	true	true	WH_PROP_BE	0.23	0.334	A2	10	Max

Table 4.22: Best results accomplished with LSA, for WWBM English corpus. The whole table may be consulted in annex (Table D.1).

For Portuguese corpus, we just used *Bing*, for the reasons regarding *Blekko* performance with the Portuguese language. Results are in Table 4.23.

Search	AQ	Quote	Filter	Acc.	C@1	App.	Passages	Norm.
BING	true	true	BE	0.29	0.342	A2	5	-
BING	true	true	BE	0.29	0.342	A2	10	-
BING	true	true	BE	0.28	0.330	A1	5	Max

Table 4.23: Best results accomplished with LSA, for WWBM Portuguese corpus. The whole table may be consulted in annex (Table D.2).

4.5.2 Question Answering for Machine Reading Evaluation

Table 4.24 presents the results for the semantic spaces approaches applied to QA4MRE. We also tried another approach using Lucene, but differently from the presented in previously sections. We extended each test document, for each question, with the top passages returned by Lucene, with the respective question being the query. The idea is to help create the matrix A, defining this way better the semantic space. In contrary, only using those passages would

[7]Unfortunately, how the framework actually works, it is not possible to calculate the MRR results.

not be sufficient to model the space, resulting in a sparse matrix. Also, is important to note that this approach would not work for the lexical approaches, as they are dependent on the the text sequence – these passages are simply appended to the document. Results are shown in Table 4.25[7].

	Freq.		**Norm. (Sum)**		**Norm. (Max)**	
App.	A1	A2	A1	A2	A1	A2
Acc.	0.192	0.258	0.175	0.267	0.192	0.242
C@1	0.212	0.286	0.194	0.296	0.212	0.268

Table 4.24: Results accomplished with LSA, for QA4MRE 2011 corpus.

	Freq.		**Norm. (Sum)**		**Norm. (Max)**	
App.	A1	A2	A1	A2	A1	A2
Acc.	0.242	0.250	0.125	0.250	0.175	0.200
C@1	0.268	0.277	0.139	0.277	0.194	0.222

Table 4.25: Results accomplished with LSA, for QA4MRE 2011 corpus, expanding the document with passages from Lucene.

4.5.3 Discussion

Looking to our third approach, not much conclusions can be drawn from the results. Regarding the two approaches (A1 and A2), it is interesting that the second performs much better, overall, in opposition to our intuition. It would make sense that, when selecting the passage more similar to the question, it would be possible to better choose the candidate answer. However, results show otherwise. In what concerns normalization factors, it seems to be much task-dependent, with none standing out.

Regarding WWBM, it is also not clear the best value for passages. Finally, it is clear that only indexing the documents together with the Background Collection, using Lucene, is not enough for the QA4MRE task.

[7]Unfortunately, how the framework actually works, it is not possible to calculate the MRR results.

4.6 Discussion

In the chapter we presented the different techniques that AnSelMo incorporates, as well as their results in three different scenarios. Results are different between them, and not standardized. In fact, the best parametrization for one scenario, in a given technique, is not the same for another scenario. An overview of the best results is in Table 4.26.

	WWBM (EN)	**WWBM (PT)**	**QA4MRE**	**Just.Ask**
Counting	0.70	0.50	-	-
Word Proximity	0.77	0.51	0.317	0.612
Extents	0.67	0.40	0.267	-
Extents Points	0.72	0.44	0.233	-
LSA	0.45	0.29	0.267	-

Table 4.26: Best results accomplished for all scenarios, with all techniques. The presented values are accuracies.

Counting technique performs as well as most techniques, but it is probably not accurate for a real QA system, as in the scenario it was applied (WWBM) candidates are not related and, thus, corresponding queries may be sufficient to choose the correct answer. On the other hand, LSA obtained the worst results, even in a scenario like QA4MRE. In WWBM we used a set of documents related with each of the candidate answers, so probably there are some latent topics that are shared by all answers, not allowing the system to distinguished them. However, in QA4MRE, we only used one document – the test document. The poor results may be related with the task nature, and, for this reason, other features to fill matrix A should be studied, such as `tf.idf` scores.

Similarity Measures had some interesting results, namely `Extents Points`, as this technique can improve significantly `Extents`' *Bing* results. However, a deeper analysis must be done, as it is not clear their impact when compared with Word Proximity.

When compared with state of the art, results obtained for WWBM are better. However, our results cannot be compared straightforward with results obtained in the beginning of this century (around 70%- 75% accuracy), as we are using different search engines and different

corpora. We also noticed that, for the aggregation of the best WWBM runs, the system could answer correctly to over 90 of the 100 questions, that is, if the techniques were perfectly combined, accuracies of 90% could be achieved. This leads to the necessity to combine the different techniques. Techniques from Learn to Rank and other combination strategies may bring significant benefits to AnSelMo.

Regarding QA4MRE, results are lower than the 2011 winning system, but they surpass almost all other competing systems. The same techniques were used to submit a few runs to 2012 track, and results are similar. In 2012, another topic ("Alzheimer") was added to the three topics already present in 2011. The results are not discussed in this thesis because the Goldstandard, at the time of writing, is not yet available, as well as other competing systems results.

For WWBM Portuguese corpus, results are lower than for English and are quite similar among the studied techniques, although with highlight for Word Proximity, once again. We believe, however, that these lower results are strongly related with the less information that can be found on the web.

Finally, Just.Ask results are promising. The best conclusion we can draw from the obtained results is that our techniques can, in fact, help a QA system improve its performance, by pushing more often the correct answer to the top.

5 Conclusions and Future Work

5.1 Conclusions

In this thesis we studied the answer selection problem in QA systems, that aims at selecting one or more candidate answers as correct answers. We propose an approach based on context; given a candidate answer, we account the text where it was extracted.

We developed AnSelMo, an answering selection module, based on some previous state of the art techniques. Besides Counting, which chooses the answer only based on the number of search engine hits, AnSelMo implements two different approaches: Lexical Distance approaches and a Semantic Space-based approach. The first includes Word Proximity, an algorithm that weighs the distance between question and answer terms, and Similarity Measures, which compares the extents found, that is, the passages that represent the question and the answers, by containing some terms from them. Latent Semantic Analysis is the Semantic Space technique used, and tries to capture latent topics from the documents. These documents are related with the candidate answers.

We used AnSelMo in three different scenarios: WWBM, QA4MRE and Just.Ask. The first two contain questions with a fixed number of candidate answers, non-related, where one is the correct answer. The later is a QA system, where questions have a different number of candidates that can be related. Results showed that the techniques can achieve similar accuracy results to the Counting technique (77% against 70% for WWBM), which is not always applicable in real scenarios, surpassing state of the art results (75% by Lam et al. [22] for WWBM scenario). In QA4MRE we were able to surpass most of the 2011 systems (by reaching 0.35 in c@1), although we are still below the winning system (0.57 in c@1). Finally, using AnSelMo as an module in Just.Ask allows the QA system to improve its results

(approximately 8 questions in 558).

5.2 Future Work

Although AnSelMo accomplished promising results in all scenarios, there is still much room for improvement. Some possible extensions are the following:

- Different Information Sources can be used. In this thesis we only used *Bing* and *Blekko* as search engines, and FreeLing POS tagger. However, the usage of WordNet and a stemmer/lemmatizer (see Section 2.6) would allow the system to identify synonym words or conjugations of verbs as the *same* word;

- Other Semantic Space models could be applied, as pLSA or LDA, as they use a probabilist model, allowing words to belong to more than a topic, even if not explicit in the initial documents, by attributing a small probability of the word belonging to such topics. Also, different strategies to create matrix A could be developed, as using `tf.idf` scores, collapse synonyms, etc.;

- The presented techniques can be more adequate to some questions than others, and may, thus, be able to answer to a different set of questions. The different ordinations could be merged into a final ordination, increasing this way the number of correct answers. Techniques that combine the returned scores should be studied, such as self-learned weights or Learn to Rank algorithms;

- The `Extents` and `Extents Points` techniques were both used to apply Similarity Measures, but they could also be used to identify the related snippets and, then, be used in Word Proximity technique. Also, other similarity measures could be used, including those that care about order.

Bibliography

[1] João Pedro Carlos Gomes da Silva. Qa+ml@wikipedia&google (Masterthesis), November 2009.

[2] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *In 6th Int'l Semantic Web Conference, Busan, Korea*, pages 11–15. Springer, 2007.

[3] Svitlana Babych, Alexander Henn, Jan Pawellek, and Sebastian Padó. Dependency-based answer validation for german. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[4] Kurt Bollacker, Patrick Tufts, Tomi Pierce, and Robert Cook. *A Platform for Scalable , Collaborative , Structured Information Integration.* 2007. URL `http://www.aaai.org/Papers/Workshops/2007/WS-07-14/WS07-14-004.pdf`.

[5] Gosse Bouma, Geert Kloosterman, Jori Mur, Gertjan van Noord, Lonneke van der Plas, and Jörg Tiedemann. Question answering with joost at clef 2007. In Peters et al. [35], pages 257–260. ISBN 978-3-540-85759-4.

[6] Ling Cao, Xipeng Qiu, and Xuanjing Huang. Question answering for machine reading with lexical chain. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[7] Adán Cassan, Helena Figueira, André Martins, Afonso Mendes, Pedro Mendes, Cláudia Pinto, and Daniel Vidal. Priberam's question answering system in a cross-language environment. In Alessandro Nardi, Carol Peters, and José L. Vicedo, editors, *Results of the CLEF 2006 Cross-Language System Evaluation Campaign, Working Notes for the CLEF 2006 Workshop*, volume 4730, pages 300–309, Alicante, Spain, 2006. Springer.

[8] Charles L. A. Clarke, Gordon V. Cormack, D. I. E. Kisman, and Thomas R. Lynam. Question answering by passage selection (multitext experiments for trec-9). In *Proceedings of TREC-2000, 9th Text Retrieval Conference*, 2000.

[9] Charles L. A. Clarke, Gordon V. Cormack, and Thomas R. Lynam. Exploiting redundancy in question answering. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '01, pages 358–365, New York, NY, USA, 2001. ACM.

[10] Scott Deerwester, Susan T. Dumais, George W. Furnas, Thomas K. Landauer, and Richard Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[11] Abdessamad Echihabi, Ulf Hermjakob, Eduard Hovy, Daniel Marcu, Eric Melz, and Deepak Ravichandran. How to select an answer string? In Tomek Strzalkowski and

Sanda Harabagiu, editors, *Advances in Textual Question Answering*. Kluwer, 2004. URL http://www.isi.edu/~marcu/papers.html.

[12] 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.

[13] Michael Fleischman, Eduard Hovy, and Abdessamad Echihabi. Offline strategies for online question answering: answering questions before they are asked. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics - Volume 1*, ACL '03, pages 1–7, Stroudsburg, PA, USA, 2003. Association for Computational Linguistics. doi: http://dx.doi.org/10.3115/1075096.1075097. URL http://dx.doi.org/10.3115/1075096.1075097.

[14] Ingo Glöckner, Björn Pelzer, and Tiansi Dong. The loganswer project at qa4mre 2011. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[15] Antonio J. Gonzalez, Alberto T. Valero, Claudia D. Carral, Manuel, and Luis V. Pineda. INAOE at CLEF 2006: Experiments in Spanish question answering. In Alessandro Nardi, Carol Peters, and José L. Vicedo, editors, *Results of the CLEF 2006 Cross-Language System Evaluation Campaign, Working Notes for the CLEF 2006 Workshop*, Alicante, Spain, 2006.

[16] Chedid Haddad and Bipin C. Desai. *Bilingual Question Answering Using CINDI_QA at QA@CLEF 2007*, pages 308–315. Springer-Verlag, Berlin, Heidelberg, 2008. ISBN 978-3-540-85759-4. doi: 10.1007/978-3-540-85760-0_37. URL http://dl.acm.org/citation.cfm?id=1428850.1428896.

[17] M. Hearst. *Automated discovery of WordNet relations*, pages 131–152. MIT Press, Cambridge, MA, 1998.

[18] Marti A. Hearst. Automatic acquisition of hyponyms from large text corpora. In *Proceedings of the 14th conference on Computational linguistics - Volume 2*, COLING '92, pages 539–545, Stroudsburg, PA, USA, 1992. Association for Computational Linguistics.

[19] Adrian Iftene, Alexandru-Lucian Gînsca, Mihai Alex Moruz, Diana Trandabat, and Maria Husarciuc. Question answering for machine reading evaluation on romanian and english languages. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[20] Hideo Joho and Mark Sanderson. Retrieving descriptive phrases from large amounts of free text. In *Proceedings of the ninth international conference on Information and knowledge management*, CIKM '00, pages 180–186, New York, NY, USA, 2000. ACM. ISBN 1-58113-320-0. doi: http://doi.acm.org/10.1145/354756.354817. URL http://doi.acm.org/10.1145/354756.354817.

[21] Julian Kupiec. Murax: a robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '93, pages 181–190, New York, NY, USA, 1993. ACM. ISBN 0-89791-605-0. doi: http://doi.acm.org/10.1145/160688.160717. URL http://doi.acm.org/10.1145/160688.160717.

[22] S.K. Lam, D.M. Pennock, D. Cosley, and S Lawrence. 1 billion pages = 1 million dollars? mining the web to play "who wants to be a millionaire?". In *Uncertainty in Artificial Intelligence (UAI2003)*, pages 337–345, Acapulco, Mexico, 2003. URL `http://www.grouplens.org/papers/pdf/1m-uai2003.pdf`.

[23] V. I. Levenshtein. Binary Codes Capable of Correcting Deletions, Insertions and Reversals. *Soviet Physics Doklady*, 10:707–710, 1966.

[24] Xin Li and Dan Roth. Learning question classifiers. In *Proceedings of the 19th international conference on Computational linguistics*, pages 1–7, Morristown, NJ, USA, 2002. Association for Computational Linguistics.

[25] Jimmy Lin. An exploration of the principles underlying redundancy-based factoid question answering. *ACM Trans. Inf. Syst.*, 25(2):6, 2007.

[26] Jimmy Lin and Boris Katz. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the twelfth international conference on Information and knowledge management*, CIKM '03, pages 116–123, New York, NY, USA, 2003. ACM. ISBN 1-58113-723-0. doi: http://doi.acm.org/10.1145/956863.956886. URL `http://doi.acm.org/10.1145/956863.956886`.

[27] Juan Martinez-Romo and Lourdes Araujo. Graph-based word clustering applied to question answering and reading comprehension tests. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[28] Ana Mendes, Luísa Coheur, and João Silva. Just.ask - a multi-pronged approach to question answering, 2012.

[29] Ana Cristina Mendes and Luísa Coheur. An approach to answer selection in question-answering based on semantic relations. In Toby Walsh, editor, *IJCAI 2011, Proceedings of the 22nd International Joint Conference on Artificial Intelligence, Barcelona, Catalonia, Spain, July 16-22, 2011*, pages 1852–1857. IJCAI/AAAI, 2011. ISBN 978-1-57735-516-8. doi: http://ijcai.org/papers11/Papers/IJCAI11-310.pdf.

[30] George A. Miller. Wordnet: a lexical database for english. *Commun. ACM*, 38:39–41, November 1995. ISSN 0001-0782. doi: http://doi.acm.org/10.1145/219717.219748. URL `http://doi.acm.org/10.1145/219717.219748`.

[31] Dan Moldovan and Adrian Novischi. Lexical chains for question answering. In *Proc. COLING 2002*, 2002. URL `http://xwn.hlt.utdallas.edu/papers.html`.

[32] Gabriel Murray, Steve Renals, and Jean Carletta. Extractive summarization of meeting recordings. In *INTERSPEECH*, pages 593–596. ISCA, 2005. URL `http://dblp.uni-trier.de/db/conf/interspeech/interspeech2005.html\#MurrayRC05`.

[33] Partha Pakray, Pinaki Bhaskar, Somnath Banerjee, Bidhan Chandra Pal, Sivaji Bandyopadhyay, and Alexander F. Gelbukh. A hybrid question answering system based on information retrieval and answer validation. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[34] Anselmo Peñas, Eduard H. Hovy, Pamela Forner, Álvaro Rodrigo, Richard F. E. Sutcliffe, Corina Forascu, and Caroline Sporleder. Overview of qa4mre at clef 2011: Question answering for machine reading evaluation. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[35] 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*, 2008. Springer. ISBN 978-3-540-85759-4.

[36] Vivien Petras, Pamela Forner, and Paul D. Clough, editors. *CLEF 2011 Labs and Workshop, Notebook Papers, 19-22 September 2011, Amsterdam, The Netherlands*, 2011. ISBN 978-88-904810-1-7.

[37] Pascal Pons and Matthieu Latapy. Computing communities in large networks using random walks. In Pinar Yolum, Tunga Güngör, Fikret S. Gürgen, and Can C. Özturan, editors, *ISCIS*, volume 3733 of *Lecture Notes in Computer Science*, pages 284–293. Springer, 2005. ISBN 3-540-29414-7.

[38] John Prager, Eric Brown, Dragomir R. Radev, and Krzysztof Czuba. One search engine or two for question-answering. In *In Nineth Text REtrieval Conference, Gaithersburg,USA*, 2000.

[39] D. Ravichandran and Eduard Hovy. Learning surface text patterns for a question answering system. In *Proceedings of the 40th ACL conference. Philadelphia, PA.*, 2002. URL `http://www.isi.edu/natural-language/projects/webclopedia/pubs/02ACL-patterns.pdf`.

[40] Ricardo Daniel Santos Faro Marques Ribeiro and David Martins de Matos. Summarizing speech by contextual reinforcement of important passages. In *PROPOR*, pages 392–402, 2012.

[41] Álvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. Uned at answer validation exercise 2007. In Peters et al. [35], pages 404–409. ISBN 978-3-540-85759-4.

[42] Álvaro Rodrigo, Anselmo Peñas, and Felisa Verdejo. Overview of the answer validation exercise 2008. In *Proceedings of the 9th Cross-language evaluation forum conference on Evaluating Systems for Multilingual and Multimodal Information Access*, CLEF'08, pages 296–313, Berlin, Heidelberg, 2009. Springer-Verlag. ISBN 3-642-04446-8, 978-3-642-04446-5. URL `http://dl.acm.org/citation.cfm?id=1813809.1813853`.

[43] Magnus Sahlgren. *The Word-Space Model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces.* PhD thesis, Stockholm University, 2006.

[44] José Saias and Paulo Quaresma. The di@ue's participation in qa4mre: from qa to multiple choice challenge. In Petras et al. [36]. ISBN 978-88-904810-1-7.

[45] Luís Sarmento, Jorge Teixeira, and Eugénio Oliveira. Experiments with query expansion in the raposa (fox) question answering system. Volume 8,

pages 792–798, 2008. URL `http://www.clef-campaign.org/2008/working_notes/sarmento-paperCLEF2008.pdf`.

[46] Nico Schlaefer, Petra Gieselman, and Guido Sautter. The ephyra qa system at trec 2006. In *Proceedings of TREC-2006, 15th Text Retrieval Conference*, 2006. URL `http://www.cs.cmu.edu/~nico/pubs/trec2006_schlaefer.pdf`.

[47] J. Silva, L. Coheur, A. Mendes, and A. Wichert. From symbolic to sub-symbolic information in question classification. *Artificial Intelligence Review*, 2010. accepted for publication.

[48] Steven Sinha and Srini Narayanan. Model-based answer selection. In *Proceedings of the Twentieth National Conference on Artificial Intelligence (AAAI-05)*, Pittsburgh, PA, USA, 2005.

[49] M. M. Soubbotin. Patterns of potential answer expressions as clues to the right answers. In *Proceedings of TREC-2001, 10th Text Retrieval Conference*, pages 293–302, 2001.

[50] Mark Steyvers and Tom Griffiths. *Probabilistic Topic Models*. Lawrence Erlbaum Associates, 2007. ISBN 1410615340. URL `http://www.amazon.com/exec/obidos/redirect?tag=citeulike07-20\&path=ASIN/1410615340`.

[51] T. Strohman, D. Metzler, H. Turtle, and W.B. Croft. Indri: A language model-based search engine for complex queries. In *Proceedings of the International Conference on Intelligent Analysis*, 2005.

[52] Fabian M. Suchanek, Gjergji Kasneci, and Gerhard Weikum. Yago: A Core of Semantic Knowledge. In *16th international World Wide Web conference (WWW 2007)*, New York, NY, USA, 2007. ACM Press.

[53] Renxu Sun, Hang Cui, Keya Li, Min-Yen Kan, and Tat-Seng Chua. Dependency relation matching for answer selection. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval*, SIGIR '05, pages 651–652, New York, NY, USA, 2005. ACM. ISBN 1-59593-034-5. doi: http://doi.acm.org/10.1145/1076034.1076173. URL `http://doi.acm.org/10.1145/1076034.1076173`.

[54] Suzan Verberne. Retrieval-based question answering for machine reading evaluation. In Petras et al. [36]. ISBN 978-88-904810-1-7.

A Counting Results

The following tables comprise the results for all runs using the Counting technique. The results are ordered by AQ and Quote, and then by Accuracy and MRR. The shaded rows contain the best runs, presented in Section 4.2.

Table A.1: Results for Counting strategy, using *Bing*, for English WWBM corpus.

Search	AQ	Quote	Filter	Accuracy	MRR	C@1
BING	true	true	WH_PROP_BE	0.70	0.802	0.714
BING	true	true	BE	0.67	0.783	0.683
BING	true	true	NO	0.66	0.783	0.673
BING	true	true	WH_PROP	0.66	0.775	0.673
BING	true	true	PROP	0.64	0.776	0.653
BING	true	true	PROP_BE	0.64	0.773	0.653
BING	true	true	WH_BE	0.63	0.763	0.643
BING	true	true	WH	0.60	0.745	0.612
BING	true	false	WH_PROP	0.59	0.729	0.602
BING	true	false	WH	0.58	0.735	0.592
BING	true	false	BE	0.58	0.731	0.592
BING	true	false	PROP_BE	0.57	0.733	0.581
BING	true	false	WH_BE	0.57	0.733	0.581
BING	true	false	WH_PROP_BE	0.57	0.721	0.581
BING	true	false	PROP	0.56	0.733	0.571
BING	true	false	NO	0.55	0.713	0.561
BING	false	true	NO	0.73	0.834	0.745
BING	false	true	BE	0.69	0.812	0.704
BING	false	true	PROP_BE	0.69	0.807	0.704
BING	false	true	WH_PROP	0.69	0.803	0.704
BING	false	true	WH	0.68	0.809	0.694
BING	false	true	PROP	0.66	0.789	0.673
BING	false	true	WH_PROP_BE	0.66	0.785	0.673
BING	false	true	WH_BE	0.62	0.766	0.632
BING	false	false	PROP	0.59	0.755	0.602
BING	false	false	WH_BE	0.58	0.737	0.592
BING	false	false	WH_PROP_BE	0.57	0.733	0.581
BING	false	false	PROP_BE	0.57	0.732	0.581
BING	false	false	WH	0.56	0.726	0.571

Table A.1 – Continued

Search	AQ	Quote	Filter	Accuracy	MRR	C@1
BING	false	false	NO	0.56	0.723	0.571
BING	false	false	WH_PROP	0.55	0.723	0.561
BING	false	false	BE	0.53	0.702	0.541

Table A.2: Results for Counting strategy, using *Blekko*, for English WWBM corpus.

Search	AQ	Quote	Filter	Accuracy	MRR	C@1
BLEKKO	true	true	WH_PROP_BE	0.67	0.802	0.723
BLEKKO	true	true	WH_PROP	0.66	0.794	0.718
BLEKKO	true	true	WH	0.65	0.787	0.680
BLEKKO	true	true	WH_BE	0.64	0.781	0.670
BLEKKO	true	true	NO	0.56	0.722	0.607
BLEKKO	true	true	PROP	0.54	0.703	0.588
BLEKKO	true	true	BE	0.51	0.696	0.570
BLEKKO	true	true	PROP_BE	0.37	0.580	0.405
BLEKKO	true	false	PROP_BE	0.62	0.767	0.650
BLEKKO	true	false	PROP	0.60	0.767	0.624
BLEKKO	true	false	BE	0.58	0.748	0.604
BLEKKO	true	false	WH_PROP	0.57	0.748	0.611
BLEKKO	true	false	NO	0.57	0.732	0.582
BLEKKO	true	false	WH_PROP_BE	0.55	0.738	0.583
BLEKKO	true	false	WH_BE	0.51	0.714	0.540
BLEKKO	true	false	WH	0.48	0.693	0.508
BLEKKO	false	true	WH_PROP_BE	0.58	0.753	0.621
BLEKKO	false	true	WH_PROP	0.54	0.718	0.583
BLEKKO	false	true	PROP	0.53	0.702	0.588
BLEKKO	false	true	PROP_BE	0.53	0.698	0.588
BLEKKO	false	true	BE	0.39	0.611	0.413
BLEKKO	false	true	WH_BE	0.39	0.608	0.448
BLEKKO	false	true	NO	0.34	0.572	0.345
BLEKKO	false	true	WH	0.33	0.562	0.339
BLEKKO	false	false	WH_PROP_BE	0.57	0.734	0.605
BLEKKO	false	false	WH_PROP	0.56	0.724	0.594
BLEKKO	false	false	PROP_BE	0.53	0.701	0.587
BLEKKO	false	false	PROP	0.52	0.695	0.555
BLEKKO	false	false	WH_BE	0.42	0.639	0.443
BLEKKO	false	false	WH	0.39	0.603	0.414
BLEKKO	false	false	BE	0.38	0.616	0.403
BLEKKO	false	false	NO	0.27	0.550	0.264

Table A.3: Results for Counting strategy, using *Bing*, for Portuguese WWBM corpus.

Search	AQ	Quote	Filter	Accuracy	MRR	C@1
BING	true	true	WH_PROP_BE	0.50	0.679	0.488
BING	true	true	WH_BE	0.50	0.677	0.486
BING	true	true	PROP_BE	0.50	0.675	0.488
BING	true	true	WH_PROP	0.49	0.674	0.477
BING	true	true	BE	0.49	0.673	0.475
BING	true	true	WH	0.49	0.672	0.475
BING	true	true	PROP	0.49	0.670	0.477
BING	true	true	NO	0.48	0.668	0.464
BING	true	false	WH_PROP_BE	0.46	0.652	0.460
BING	true	false	WH_PROP	0.46	0.653	0.460
BING	true	false	WH_BE	0.42	0.634	0.416
BING	true	false	PROP_BE	0.43	0.631	0.428
BING	true	false	WH	0.42	0.634	0.416
BING	true	false	PROP	0.43	0.633	0.428
BING	true	false	BE	0.43	0.633	0.426
BING	true	false	NO	0.42	0.628	0.416
BING	false	true	WH_BE	0.49	0.678	0.475
BING	false	true	BE	0.49	0.673	0.475
BING	false	true	WH_PROP	0.49	0.670	0.477
BING	false	true	PROP	0.49	0.668	0.477
BING	false	true	WH_PROP_BE	0.48	0.668	0.466
BING	false	true	NO	0.48	0.666	0.471
BING	false	true	PROP_BE	0.48	0.665	0.466
BING	false	true	WH	0.47	0.666	0.460
BING	false	false	WH_PROP_BE	0.41	0.623	0.407
BING	false	false	WH_PROP	0.41	0.626	0.407
BING	false	false	WH_BE	0.44	0.648	0.437
BING	false	false	PROP_BE	0.42	0.624	0.417
BING	false	false	WH	0.45	0.650	0.447
BING	false	false	PROP	0.42	0.627	0.417
BING	false	false	BE	0.42	0.627	0.416
BING	false	false	NO	0.43	0.629	0.426

Table A.4: Results for Counting strategy, using *Blekko*, for Portuguese WWBM corpus.

Search	AQ	Quote	Filter	Accuracy	MRR	C@1
BLEKKO	true	true	WH_PROP_BE	0.31	0.565	0.237
BLEKKO	true	true	WH_PROP	0.29	0.555	0.220
BLEKKO	true	true	PROP_BE	0.29	0.554	0.203
BLEKKO	true	true	WH_BE	0.28	0.551	0.237
BLEKKO	true	true	BE	0.28	0.543	0.245
BLEKKO	true	true	PROP	0.27	0.544	0.186
BLEKKO	true	true	WH	0.26	0.540	0.207
BLEKKO	true	true	NO	0.26	0.534	0.216
BLEKKO	true	false	WH_PROP_BE	0.30	0.558	0.228
BLEKKO	true	false	NO	0.30	0.556	0.277
BLEKKO	true	false	BE	0.30	0.553	0.276
BLEKKO	true	false	WH_PROP	0.29	0.553	0.213
BLEKKO	true	false	PROP_BE	0.28	0.546	0.197
BLEKKO	true	false	WH_BE	0.28	0.538	0.237
BLEKKO	true	false	PROP	0.27	0.541	0.182
BLEKKO	true	false	WH	0.27	0.532	0.224
BLEKKO	false	true	WH_PROP_BE	0.30	0.557	0.225
BLEKKO	false	true	WH_BE	0.29	0.548	0.283
BLEKKO	false	true	PROP_BE	0.28	0.548	0.175
BLEKKO	false	true	WH_PROP	0.28	0.544	0.193
BLEKKO	false	true	WH	0.28	0.543	0.268
BLEKKO	false	true	PROP	0.27	0.540	0.158
BLEKKO	false	true	NO	0.26	0.525	0.250
BLEKKO	false	true	BE	0.26	0.525	0.252
BLEKKO	false	false	WH_PROP_BE	0.31	0.563	0.246
BLEKKO	false	false	WH_PROP	0.30	0.554	0.216
BLEKKO	false	false	WH_BE	0.26	0.530	0.243
BLEKKO	false	false	PROP_BE	0.28	0.544	0.184
BLEKKO	false	false	WH	0.26	0.530	0.243
BLEKKO	false	false	PROP	0.28	0.545	0.169
BLEKKO	false	false	BE	0.26	0.530	0.245
BLEKKO	false	false	NO	0.27	0.533	0.257

B Word Proximity Results

The following table (Table B.1) presents the results for the 216 tests, which vary the radius values (20, 40 and 60) and the number of passages used (5, 10 and 20). Shaded rows contain the best results, presented in Section 4.3. The remaining tables present other results obtained with Word Proximity. Results are ordered by Accuracy, MRR and c@1. Shaded rows contain the best results, presented in Section 4.3.

Table B.1: Results for Word Proximity strategy, for English WWBM corpus.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BLEKKO	true	false	WH_PROP_BE	0.68	0.806	0.730	20	20
BLEKKO	true	false	WH_PROP_BE	0.68	0.805	0.720	20	5
BLEKKO	true	true	WH_PROP_BE	0.67	0.803	0.725	20	20
BLEKKO	true	false	WH_PROP_BE	0.67	0.798	0.717	40	20
BLEKKO	true	true	WH_PROP_BE	0.66	0.798	0.708	40	20
BLEKKO	true	true	WH_PROP_BE	0.66	0.796	0.713	60	20
BLEKKO	true	false	WH_PROP_BE	0.66	0.790	0.720	20	10
BLEKKO	true	false	WH_PROP_BE	0.66	0.790	0.696	40	5
BLEKKO	true	true	WH_PROP_BE	0.66	0.789	0.732	20	10
BLEKKO	true	false	WH_PROP	0.66	0.789	0.696	20	5
BLEKKO	true	true	WH_BE	0.66	0.785	0.706	60	20
BLEKKO	true	false	WH_PROP_BE	0.65	0.782	0.708	40	10
BING	false	true	PROP_BE	0.65	0.772	0.663	20	20
BING	true	true	BE	0.65	0.771	0.663	60	20
BLEKKO	true	true	WH_PROP	0.64	0.783	0.689	20	5
BLEKKO	true	false	WH_PROP_BE	0.64	0.779	0.696	60	10
BLEKKO	true	true	WH_PROP	0.64	0.779	0.684	20	20
BLEKKO	true	false	WH_PROP	0.64	0.776	0.672	40	5
BLEKKO	true	true	WH_PROP	0.64	0.775	0.702	60	5
BLEKKO	true	true	WH_PROP	0.64	0.775	0.678	40	5
BLEKKO	true	true	WH_BE	0.64	0.774	0.672	40	20
BLEKKO	true	true	WH_PROP	0.64	0.773	0.684	60	20
BING	true	true	WH_PROP_BE	0.64	0.763	0.653	40	20
BING	true	true	WH_PROP_BE	0.64	0.763	0.653	60	20
BING	false	true	WH_PROP	0.64	0.760	0.653	20	20
BLEKKO	true	true	WH_PROP_BE	0.63	0.780	0.677	20	5
BLEKKO	true	false	WH_PROP	0.63	0.780	0.673	40	20
BLEKKO	true	false	WH_PROP	0.63	0.772	0.684	20	10

Table B.1 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BLEKKO	true	true	WH_BE	0.63	0.771	0.684	40	10
BLEKKO	true	true	WH_PROP	0.63	0.771	0.673	40	20
BLEKKO	true	false	WH_PROP	0.63	0.770	0.684	40	10
BLEKKO	true	true	WH_BE	0.63	0.768	0.661	20	20
BLEKKO	true	true	WH_PROP	0.63	0.767	0.684	20	10
BING	false	true	PROP_BE	0.63	0.763	0.643	60	20
BING	true	true	WH_BE	0.63	0.763	0.643	60	20
BING	true	true	WH	0.63	0.763	0.643	60	20
BING	false	true	PROP_BE	0.63	0.763	0.643	40	20
BING	true	true	PROP	0.63	0.758	0.643	40	20
BING	true	true	NO	0.63	0.758	0.643	60	20
BLEKKO	true	false	WH_PROP	0.62	0.773	0.667	20	20
BLEKKO	true	false	WH_PROP_BE	0.62	0.772	0.661	60	20
BLEKKO	true	true	WH_PROP_BE	0.62	0.770	0.673	40	10
BLEKKO	true	true	WH_PROP_BE	0.62	0.770	0.679	60	10
BLEKKO	true	false	WH_PROP_BE	0.62	0.770	0.673	60	5
BLEKKO	true	false	WH_PROP	0.62	0.768	0.678	60	10
BLEKKO	true	true	WH_BE	0.62	0.765	0.679	60	10
BLEKKO	true	true	WH_BE	0.62	0.763	0.673	20	10
BING	true	true	WH_BE	0.62	0.754	0.632	40	20
BING	true	true	WH	0.62	0.754	0.632	40	20
BING	false	true	PROP	0.62	0.753	0.632	20	20
BING	true	true	WH_PROP	0.62	0.753	0.632	40	20
BING	false	true	WH_PROP	0.62	0.753	0.632	40	20
BING	false	true	WH_PROP	0.62	0.753	0.632	60	20
BING	true	true	PROP_BE	0.62	0.753	0.632	60	20
BING	true	true	PROP	0.62	0.753	0.632	60	20
BING	true	true	WH_PROP	0.62	0.753	0.632	60	20
BING	true	true	WH	0.62	0.752	0.632	60	10
BING	true	true	NO	0.62	0.749	0.632	40	20
BLEKKO	true	true	WH_PROP_BE	0.61	0.767	0.648	40	5
BLEKKO	true	false	WH_BE	0.61	0.765	0.649	20	20
BLEKKO	true	true	WH_PROP_BE	0.61	0.763	0.678	60	5
BLEKKO	true	false	WH_BE	0.61	0.758	0.655	40	10
BLEKKO	true	true	WH_PROP	0.61	0.755	0.660	60	10
BING	false	true	WH	0.61	0.751	0.622	60	20
BING	true	true	BE	0.61	0.751	0.622	40	20
BING	false	true	WH	0.61	0.751	0.622	40	20
BING	false	true	PROP	0.61	0.749	0.622	60	20
BING	false	true	PROP	0.61	0.749	0.622	40	20
BLEKKO	true	false	PROP_BE	0.61	0.748	0.663	20	10
BLEKKO	true	false	PROP_BE	0.61	0.748	0.662	40	10
BING	false	true	BE	0.61	0.748	0.622	60	20
BING	false	true	BE	0.61	0.748	0.622	40	20

Table B.1 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	true	true	PROP_BE	0.61	0.748	0.622	40	20
BING	true	true	WH_BE	0.61	0.743	0.622	60	10
BLEKKO	true	false	WH_BE	0.60	0.764	0.649	60	20
BLEKKO	true	false	WH_PROP	0.60	0.763	0.644	60	20
BLEKKO	true	false	WH_PROP	0.60	0.756	0.655	60	5
BLEKKO	true	true	WH_BE	0.60	0.754	0.636	20	5
BLEKKO	true	false	WH_BE	0.60	0.754	0.649	60	10
BLEKKO	true	true	WH_PROP	0.60	0.750	0.648	40	10
BING	false	true	PROP_BE	0.60	0.744	0.612	60	10
BLEKKO	true	false	PROP_BE	0.60	0.743	0.653	20	5
BLEKKO	true	false	PROP_BE	0.60	0.742	0.649	20	20
BING	true	true	WH_PROP_BE	0.60	0.740	0.612	20	20
BING	false	true	WH_PROP_BE	0.60	0.739	0.612	60	20
BING	false	true	WH_PROP_BE	0.60	0.739	0.612	40	20
BING	false	true	WH	0.60	0.738	0.612	20	20
BING	false	true	BE	0.60	0.736	0.612	60	10
BING	true	true	BE	0.60	0.735	0.612	60	10
BING	false	true	BE	0.60	0.735	0.612	40	10
BLEKKO	true	false	WH_BE	0.59	0.750	0.633	20	10
BLEKKO	true	true	WH_BE	0.59	0.748	0.620	40	5
BING	true	true	WH_PROP	0.59	0.738	0.602	60	10
BLEKKO	true	false	PROP_BE	0.59	0.737	0.630	40	20
BING	true	true	WH	0.59	0.737	0.602	20	20
BING	true	true	WH_PROP_BE	0.59	0.735	0.602	60	10
BING	true	true	WH	0.59	0.735	0.602	40	10
BING	false	true	PROP_BE	0.59	0.735	0.602	40	10
BING	false	true	BE	0.59	0.733	0.602	20	20
BING	false	true	WH_PROP_BE	0.59	0.732	0.602	20	20
BING	false	true	PROP	0.59	0.732	0.602	40	10
BING	false	true	WH	0.59	0.731	0.602	60	10
BING	false	true	NO	0.59	0.731	0.602	60	20
BING	false	true	WH_BE	0.59	0.731	0.602	60	20
BING	false	true	NO	0.59	0.731	0.602	40	20
BING	false	true	WH_BE	0.59	0.731	0.602	40	20
BING	false	true	NO	0.59	0.730	0.602	20	20
BING	false	true	WH_BE	0.59	0.730	0.602	20	20
BING	true	true	WH_BE	0.59	0.730	0.602	40	10
BING	true	true	PROP	0.59	0.730	0.602	40	10
BING	false	true	WH_PROP_BE	0.59	0.728	0.602	40	10
BING	true	true	PROP_BE	0.59	0.727	0.602	40	10
BING	false	true	WH	0.59	0.727	0.602	40	10
BLEKKO	true	false	PROP_BE	0.58	0.732	0.626	40	5
BING	false	true	PROP	0.58	0.731	0.592	60	10
BING	true	true	WH_BE	0.58	0.730	0.592	20	20

Table B.1 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	true	true	PROP_BE	0.58	0.730	0.592	60	10
BLEKKO	true	false	PROP_BE	0.58	0.729	0.639	60	10
BING	false	true	WH_PROP_BE	0.58	0.728	0.592	60	10
BING	true	true	WH_PROP	0.58	0.727	0.592	20	20
BING	true	true	NO	0.58	0.725	0.592	40	10
BING	false	true	WH_PROP	0.58	0.723	0.592	40	10
BING	true	true	BE	0.58	0.722	0.592	40	10
BING	false	true	NO	0.58	0.719	0.592	60	10
BING	false	true	WH_BE	0.58	0.719	0.592	60	10
BING	false	true	NO	0.58	0.717	0.592	40	10
BLEKKO	true	false	WH_BE	0.57	0.745	0.605	40	20
BLEKKO	true	true	WH_BE	0.57	0.732	0.626	60	5
BING	false	true	BE	0.57	0.727	0.581	60	5
BING	true	true	PROP_BE	0.57	0.727	0.581	20	20
BING	false	true	WH_PROP	0.57	0.726	0.581	60	10
BLEKKO	true	false	PROP_BE	0.57	0.724	0.621	60	20
BING	true	true	NO	0.57	0.722	0.581	20	20
BING	true	true	WH_PROP	0.57	0.720	0.581	40	10
BING	true	true	WH_PROP_BE	0.57	0.717	0.581	40	10
BING	false	true	WH_BE	0.57	0.713	0.581	40	10
BING	true	true	PROP	0.56	0.723	0.571	60	10
BING	true	true	PROP	0.56	0.722	0.571	20	20
BLEKKO	true	false	PROP_BE	0.56	0.721	0.617	60	5
BING	true	true	NO	0.56	0.720	0.571	60	10
BING	false	true	WH_PROP	0.56	0.715	0.571	60	5
BING	false	true	WH	0.55	0.717	0.561	60	5
BING	true	true	BE	0.55	0.710	0.561	20	20
BLEKKO	true	false	WH_BE	0.54	0.720	0.575	20	5
BLEKKO	true	false	WH_BE	0.54	0.712	0.570	40	5
BING	false	true	WH_PROP_BE	0.54	0.703	0.551	60	5
BING	false	true	PROP_BE	0.54	0.703	0.551	60	5
BING	false	true	WH_BE	0.52	0.695	0.530	60	5
BING	false	true	PROP	0.52	0.693	0.530	60	5
BING	false	true	WH_PROP_BE	0.52	0.688	0.530	40	5
BING	false	true	WH_PROP	0.52	0.687	0.530	40	5
BING	false	true	WH_PROP	0.52	0.685	0.530	20	10
BING	false	true	WH_PROP_BE	0.52	0.683	0.530	20	10
BLEKKO	true	false	WH_BE	0.51	0.696	0.565	60	5
BING	false	true	NO	0.51	0.692	0.520	60	5
BING	false	true	BE	0.51	0.690	0.520	40	5
BING	false	true	PROP_BE	0.51	0.688	0.520	20	10
BING	false	true	WH	0.51	0.687	0.520	40	5
BING	false	true	PROP	0.51	0.683	0.520	20	10
BING	false	true	BE	0.51	0.680	0.520	20	10

Table B.1 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	true	true	WH_BE	0.51	0.680	0.520	20	10
BING	true	true	WH	0.51	0.680	0.520	20	10
BING	true	true	PROP	0.51	0.679	0.520	20	10
BING	true	true	NO	0.51	0.677	0.520	20	10
BING	false	true	WH_BE	0.51	0.675	0.520	20	10
BING	false	true	WH	0.51	0.675	0.520	20	10
BING	false	true	PROP_BE	0.50	0.682	0.510	40	5
BING	false	true	PROP	0.50	0.678	0.510	40	5
BING	false	true	NO	0.50	0.672	0.510	20	10
BING	false	true	NO	0.49	0.675	0.500	40	5
BING	false	true	WH_BE	0.49	0.673	0.500	40	5
BING	true	true	PROP_BE	0.49	0.669	0.500	20	10
BING	true	true	WH_PROP_BE	0.49	0.667	0.500	20	10
BING	true	true	WH_PROP	0.49	0.667	0.500	20	10
BING	true	true	BE	0.49	0.660	0.500	20	10
BING	true	true	BE	0.48	0.666	0.490	60	5
BING	true	true	WH_PROP_BE	0.48	0.663	0.490	40	5
BING	true	true	PROP_BE	0.47	0.663	0.479	60	5
BING	true	true	WH	0.47	0.663	0.479	60	5
BING	true	true	WH_PROP_BE	0.47	0.663	0.479	60	5
BING	true	true	WH_PROP	0.47	0.658	0.479	60	5
BING	true	true	PROP	0.46	0.658	0.469	60	5
BING	true	true	WH_BE	0.46	0.658	0.469	60	5
BING	true	true	NO	0.46	0.654	0.469	60	5
BING	true	true	BE	0.46	0.653	0.469	40	5
BING	true	true	WH_BE	0.46	0.652	0.469	40	5
BING	true	true	WH	0.46	0.652	0.469	40	5
BING	true	true	WH_PROP	0.46	0.650	0.469	40	5
BING	true	true	PROP_BE	0.45	0.651	0.459	40	5
BING	true	true	PROP	0.45	0.651	0.459	40	5
BING	true	true	NO	0.45	0.648	0.459	40	5
BING	true	true	WH_PROP_BE	0.43	0.633	0.443	20	5
BING	true	true	PROP_BE	0.41	0.624	0.422	20	5
BING	true	true	PROP	0.41	0.624	0.422	20	5
BING	true	true	WH_BE	0.41	0.622	0.422	20	5
BING	true	true	WH	0.41	0.622	0.422	20	5
BING	true	true	WH_PROP	0.41	0.620	0.422	20	5
BING	false	true	PROP_BE	0.40	0.624	0.412	20	5
BING	false	true	PROP	0.40	0.621	0.412	20	5
BING	false	true	WH_PROP	0.40	0.621	0.412	20	5
BING	false	true	NO	0.40	0.619	0.412	20	5
BING	true	true	NO	0.40	0.617	0.412	20	5
BING	false	true	WH	0.39	0.618	0.402	20	5
BING	true	true	BE	0.39	0.615	0.402	20	5

Table B.1 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	false	true	WH_BE	0.39	0.614	0.402	20	5
BING	false	true	WH_PROP_BE	0.38	0.611	0.391	20	5
BING	false	true	BE	0.37	0.608	0.381	20	5
BLEKKO	true	true	PROP_BE	0.33	0.551	0.278	20	20
BLEKKO	true	true	PROP_BE	0.33	0.550	0.282	20	5
BLEKKO	true	true	PROP_BE	0.32	0.546	0.275	60	20
BLEKKO	true	true	PROP_BE	0.32	0.545	0.278	60	5
BLEKKO	true	true	PROP_BE	0.32	0.545	0.275	40	20
BLEKKO	true	true	PROP_BE	0.32	0.543	0.278	40	5
BLEKKO	true	true	PROP_BE	0.32	0.543	0.263	20	10
BLEKKO	true	true	PROP_BE	0.31	0.538	0.260	40	10
BLEKKO	true	true	PROP_BE	0.31	0.538	0.260	60	10

Table B.2: Results for Word Proximity strategy, for English WWBM corpus, with different radius values.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BING	true	true	BE	0.67	0.783	0.683	70	20
BING	true	true	BE	0.65	0.774	0.663	80	20
BING	false	true	PROP_BE	0.65	0.772	0.663	30	20
BING	false	true	PROP_BE	0.64	0.772	0.653	80	20
BING	false	true	PROP_BE	0.64	0.770	0.653	70	20
BING	false	true	PROP_BE	0.62	0.758	0.632	50	20
BING	true	true	BE	0.62	0.756	0.632	50	20
BING	false	true	PROP_BE	0.59	0.767	0.609	3	20
BING	true	true	BE	0.59	0.733	0.602	30	20
BING	false	true	PROP_BE	0.57	0.741	0.588	4	20
BING	false	true	PROP_BE	0.57	0.737	0.587	5	20
BING	true	true	BE	0.56	0.741	0.594	3	20
BING	false	true	PROP_BE	0.56	0.733	0.571	6	20
BING	false	true	PROP_BE	0.56	0.726	0.571	10	20
BING	true	true	BE	0.55	0.708	0.561	10	20
BING	true	true	BE	0.54	0.726	0.557	4	20
BING	true	true	BE	0.53	0.718	0.546	5	20
BING	true	true	BE	0.51	0.708	0.520	6	20
BLEKKO	true	true	WH_PROP_BE	0.76	0.849	0.842	5	20
BLEKKO	true	true	WH_PROP_BE	0.75	0.842	0.830	4	20
BLEKKO	true	true	WH_PROP_BE	0.72	0.833	0.809	6	20
BLEKKO	true	false	WH_PROP_BE	0.72	0.823	0.773	30	20
BLEKKO	true	true	WH_PROP_BE	0.71	0.827	0.777	10	20
BLEKKO	true	true	WH_PROP_BE	0.68	0.804	0.736	70	20
BLEKKO	true	true	WH_PROP_BE	0.67	0.801	0.725	80	20

Table B.2 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Rad.	Passages
BLEKKO	true	true	WH_PROP_BE	0.67	0.801	0.731	30	20
BLEKKO	true	true	WH_PROP_BE	0.67	0.801	0.725	50	20
BLEKKO	true	false	WH_PROP_BE	0.66	0.799	0.712	10	20
BLEKKO	true	false	WH_PROP_BE	0.66	0.794	0.738	4	20
BLEKKO	true	false	WH_PROP_BE	0.66	0.794	0.726	5	20
BLEKKO	true	true	WH_PROP_BE	0.66	0.783	0.728	3	20
BLEKKO	true	false	WH_PROP_BE	0.65	0.789	0.694	50	20
BLEKKO	true	false	WH_PROP_BE	0.65	0.782	0.694	80	20
BLEKKO	true	false	WH_PROP_BE	0.65	0.782	0.694	70	20
BLEKKO	true	false	WH_PROP_BE	0.64	0.788	0.708	6	20
BLEKKO	true	false	WH_PROP_BE	0.58	0.742	0.624	3	20

C Extents Results

The following tables present the results obtained with Similarity Measures. Results are ordered by Accuracy, MRR and c@1. Shaded rows contain the best results, presented in Section 4.4.

Table C.1: Results for Extents strategy, for English WWBM corpus.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.53	0.705	0.541	Dice	10
BING	true	true	BE	0.52	0.704	0.530	Dice	20
BING	false	true	PROP_BE	0.52	0.703	0.530	Jaccard	10
BING	true	true	BE	0.52	0.698	0.530	Jaccard	20
BING	false	true	PROP_BE	0.51	0.703	0.520	Dice	20
BING	true	true	BE	0.50	0.693	0.510	Jaccard	10
BING	true	true	BE	0.50	0.688	0.510	Dice	10
BING	false	true	PROP_BE	0.49	0.678	0.500	Jaccard	20
BING	true	true	BE	0.47	0.679	0.479	Dice	5
BING	true	true	BE	0.47	0.674	0.479	Jaccard	5
BING	false	true	PROP_BE	0.44	0.657	0.449	Dice	5
BING	false	true	PROP_BE	0.44	0.655	0.449	Jaccard	5
BING	false	true	PROP_BE	0.42	0.631	0.428	Overlap	10
BING	true	true	BE	0.42	0.619	0.428	Overlap	20
BING	true	true	BE	0.41	0.619	0.418	Overlap	5
BING	false	true	PROP_BE	0.41	0.618	0.418	Overlap	5
BING	false	true	PROP_BE	0.41	0.616	0.418	Overlap	20
BING	true	true	BE	0.39	0.600	0.398	Overlap	10
BLEKKO	true	true	WH_PROP_BE	0.67	0.753	0.764	Dice	20
BLEKKO	true	true	WH_PROP_BE	0.67	0.752	0.764	Jaccard	20
BLEKKO	true	true	WH_PROP_BE	0.61	0.722	0.695	Dice	5
BLEKKO	true	true	WH_PROP_BE	0.61	0.718	0.695	Dice	10
BLEKKO	true	true	WH_PROP_BE	0.61	0.717	0.695	Jaccard	10
BLEKKO	true	true	WH_PROP_BE	0.60	0.720	0.684	Jaccard	5
BLEKKO	true	false	WH_PROP_BE	0.57	0.700	0.633	Jaccard	5
BLEKKO	true	false	WH_PROP_BE	0.57	0.698	0.633	Dice	5
BLEKKO	true	false	WH_PROP_BE	0.54	0.679	0.599	Dice	20
BLEKKO	true	false	WH_PROP_BE	0.52	0.674	0.577	Dice	10
BLEKKO	true	false	WH_PROP_BE	0.52	0.663	0.577	Jaccard	20
BLEKKO	true	false	WH_PROP_BE	0.50	0.664	0.555	Jaccard	10

Table C.1 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BLEKKO	true	true	WH_PROP_BE	0.48	0.648	0.547	Overlap	20
BLEKKO	true	true	WH_PROP_BE	0.48	0.632	0.547	Overlap	5
BLEKKO	true	false	WH_PROP_BE	0.42	0.613	0.466	Overlap	5
BLEKKO	true	true	WH_PROP_BE	0.42	0.608	0.479	Overlap	10
BLEKKO	true	false	WH_PROP_BE	0.40	0.607	0.444	Overlap	20
BLEKKO	true	false	WH_PROP_BE	0.36	0.586	0.400	Overlap	10

Table C.2: Results for Extents Points strategy, for English WWBM corpus.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.72	0.817	0.763	Dice	20
BING	true	true	BE	0.68	0.791	0.721	Dice	20
BING	false	true	PROP_BE	0.66	0.787	0.700	Jaccard	20
BING	false	true	PROP_BE	0.65	0.780	0.689	Dice	5
BING	false	true	PROP_BE	0.65	0.780	0.689	Jaccard	5
BING	false	true	PROP_BE	0.65	0.780	0.689	Dice	10
BING	false	true	PROP_BE	0.63	0.770	0.668	Jaccard	10
BING	true	true	BE	0.63	0.766	0.668	Dice	5
BING	true	true	BE	0.63	0.766	0.668	Jaccard	20
BING	true	true	BE	0.62	0.760	0.657	Jaccard	5
BING	false	true	PROP_BE	0.62	0.737	0.657	Overlap	10
BING	true	true	BE	0.61	0.753	0.647	Dice	10
BING	true	true	BE	0.59	0.743	0.625	Jaccard	10
BING	true	true	BE	0.55	0.705	0.583	Overlap	5
BING	false	true	PROP_BE	0.54	0.693	0.572	Overlap	5
BING	false	true	PROP_BE	0.53	0.698	0.562	Overlap	20
BING	true	true	BE	0.51	0.693	0.541	Overlap	10
BING	true	true	BE	0.49	0.673	0.519	Overlap	20
BLEKKO	true	false	WH_PROP_BE	0.6	0.706	0.690	Dice	20
BLEKKO	true	false	WH_PROP_BE	0.59	0.696	0.679	Dice	10
BLEKKO	true	true	WH_PROP_BE	0.59	0.688	0.696	Dice	20
BLEKKO	true	true	WH_PROP_BE	0.59	0.687	0.696	Jaccard	20
BLEKKO	true	false	WH_PROP_BE	0.58	0.690	0.667	Jaccard	10
BLEKKO	true	false	WH_PROP_BE	0.57	0.691	0.656	Jaccard	20
BLEKKO	true	false	WH_PROP_BE	0.57	0.681	0.661	Dice	5
BLEKKO	true	false	WH_PROP_BE	0.57	0.681	0.661	Jaccard	5
BLEKKO	true	true	WH_PROP_BE	0.56	0.665	0.661	Dice	10
BLEKKO	true	true	WH_PROP_BE	0.56	0.663	0.661	Jaccard	10
BLEKKO	true	true	WH_PROP_BE	0.54	0.665	0.637	Dice	5
BLEKKO	true	true	WH_PROP_BE	0.53	0.660	0.625	Jaccard	5
BLEKKO	true	true	WH_PROP_BE	0.48	0.625	0.566	Overlap	20

Table C.2 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BLEKKO	true	true	WH_PROP_BE	0.44	0.593	0.519	Overlap	10
BLEKKO	true	false	WH_PROP_BE	0.43	0.591	0.495	Overlap	20
BLEKKO	true	true	WH_PROP_BE	0.43	0.585	0.507	Overlap	5
BLEKKO	true	false	WH_PROP_BE	0.35	0.534	0.406	Overlap	5
BLEKKO	true	false	WH_PROP_BE	0.34	0.539	0.391	Overlap	10

Table C.3: Results for Extents strategy, for Portuguese WWBM corpus.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.40	0.575	0.448	Jaccard	5
BING	true	true	BE	0.39	0.579	0.425	Dice	10
BING	true	true	BE	0.39	0.579	0.425	Jaccard	10
BING	false	true	PROP_BE	0.39	0.574	0.437	Jaccard	10
BING	false	true	PROP_BE	0.39	0.572	0.437	Dice	5
BING	false	true	PROP_BE	0.39	0.572	0.437	Dice	10
BING	false	true	PROP_BE	0.38	0.569	0.426	Jaccard	20
BING	false	true	PROP_BE	0.38	0.567	0.426	Dice	20
BING	true	true	BE	0.37	0.569	0.403	Dice	20
BING	true	true	BE	0.36	0.564	0.392	Jaccard	20
BING	true	true	BE	0.36	0.563	0.392	Jaccard	5
BING	true	true	BE	0.34	0.551	0.371	Dice	5
BING	true	true	BE	0.33	0.538	0.360	Overlap	20
BING	false	true	PROP_BE	0.33	0.528	0.370	Overlap	20
BING	false	true	PROP_BE	0.31	0.519	0.347	Overlap	10
BING	true	true	BE	0.29	0.513	0.316	Overlap	10
BING	false	true	PROP_BE	0.29	0.500	0.325	Overlap	5
BING	true	true	BE	0.28	0.508	0.305	Overlap	5
BLEKKO	true	false	WH_PROP_BE	0.11	0.195	0.184	Dice	20
BLEKKO	true	false	WH_PROP_BE	0.11	0.195	0.184	Jaccard	20
BLEKKO	true	false	WH_PROP_BE	0.11	0.195	0.184	Overlap	20
BLEKKO	true	false	WH_PROP_BE	0.11	0.194	0.184	Dice	10
BLEKKO	true	false	WH_PROP_BE	0.11	0.194	0.184	Jaccard	10
BLEKKO	true	false	WH_PROP_BE	0.11	0.193	0.184	Dice	5
BLEKKO	true	false	WH_PROP_BE	0.11	0.193	0.184	Jaccard	5
BLEKKO	true	true	WH_PROP_BE	0.11	0.174	0.189	Overlap	5
BLEKKO	true	true	WH_PROP_BE	0.11	0.174	0.189	Overlap	10
BLEKKO	true	true	WH_PROP_BE	0.11	0.174	0.189	Overlap	20
BLEKKO	true	false	WH_PROP_BE	0.10	0.191	0.167	Overlap	5
BLEKKO	true	false	WH_PROP_BE	0.10	0.191	0.167	Overlap	10
BLEKKO	true	true	WH_PROP_BE	0.10	0.169	0.172	Dice	20
BLEKKO	true	true	WH_PROP_BE	0.10	0.169	0.172	Jaccard	20

Table C.3 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BLEKKO	true	true	WH_PROP_BE	0.10	0.168	0.172	Dice	5
BLEKKO	true	true	WH_PROP_BE	0.10	0.168	0.172	Jaccard	5
BLEKKO	true	true	WH_PROP_BE	0.10	0.168	0.172	Dice	10
BLEKKO	true	true	WH_PROP_BE	0.10	0.168	0.172	Jaccard	10

Table C.4: Results for Extents Points strategy, for Portuguese WWBM corpus.

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BING	false	true	PROP_BE	0.44	0.577	0.519	Dice	20
BING	false	true	PROP_BE	0.44	0.577	0.519	Jaccard	20
BING	true	true	BE	0.43	0.578	0.495	Dice	5
BING	true	true	BE	0.43	0.573	0.495	Dice	10
BING	false	true	PROP_BE	0.43	0.568	0.507	Dice	5
BING	false	true	PROP_BE	0.43	0.568	0.507	Jaccard	5
BING	false	true	PROP_BE	0.43	0.568	0.507	Dice	10
BING	false	true	PROP_BE	0.43	0.568	0.507	Jaccard	10
BING	true	true	BE	0.42	0.573	0.483	Jaccard	5
BING	true	true	BE	0.42	0.570	0.483	Jaccard	20
BING	true	true	BE	0.42	0.568	0.483	Jaccard	10
BING	true	true	BE	0.41	0.565	0.472	Dice	20
BING	false	true	PROP_BE	0.37	0.526	0.437	Overlap	5
BING	true	true	BE	0.33	0.505	0.380	Overlap	5
BING	false	true	PROP_BE	0.29	0.479	0.342	Overlap	10
BING	true	true	BE	0.27	0.479	0.311	Overlap	20
BING	true	true	BE	0.27	0.478	0.311	Overlap	10
BING	false	true	PROP_BE	0.26	0.466	0.307	Overlap	20
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Dice	20
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Jaccard	20
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Overlap	20
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Overlap	5
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Dice	10
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Jaccard	10
BLEKKO	true	false	WH_PROP_BE	0.06	0.125	0.106	Overlap	10
BLEKKO	true	false	WH_PROP_BE	0.05	0.120	0.088	Dice	5
BLEKKO	true	false	WH_PROP_BE	0.05	0.120	0.088	Jaccard	5
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Dice	5
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Jaccard	5
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Overlap	5
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Dice	10
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Jaccard	10
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Overlap	10

Table C.4 – Continued

Search	AQ	Quote	Filter	Acc.	MRR	C@1	Sim.	Passages
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Dice	20
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Jaccard	20
BLEKKO	true	true	WH_PROP_BE	0.04	0.086	0.073	Overlap	20

D LSA Results

The following tables present the results obtained with LSA. Results are ordered by Accuracy, MRR and c@1. Shaded rows contain the best results, presented in Section 4.5.

Table D.1: Results for LSA, for English WWBM corpus.

Search	AQ	Quote	Filter	Acc.	C@1	App.	Passages	Norm.
BING	true	true	BE	0.45	0.504	A2	10	Max
BING	true	true	BE	0.40	0.444	A2	5	Max
BING	true	true	BE	0.37	0.444	A2	1	-
BING	true	true	BE	0.37	0.411	A1	5	Sum
BING	true	true	BE	0.34	0.408	A1	1	-
BING	true	true	BE	0.33	0.396	A2	1	Max
BING	true	true	BE	0.32	0.384	A2	1	Sum
BING	true	true	BE	0.31	0.347	A1	10	Max
BING	true	true	BE	0.31	0.344	A2	5	Sum
BING	true	true	BE	0.30	0.333	A1	5	Max
BING	true	true	BE	0.29	0.348	A1	1	Max
BING	true	true	BE	0.29	0.325	A1	10	Sum
BING	true	true	BE	0.27	0.324	A1	1	Sum
BING	true	true	BE	0.25	0.275	A1	5	-
BING	true	true	BE	0.25	0.270	A1	10	-
BING	true	true	BE	0.18	0.198	A2	5	-
BING	true	true	BE	0.17	0.190	A2	10	Sum
BING	true	true	BE	0.16	0.173	A2	10	-
BLEKKO	true	true	WH_PROP_BE	0.28	0.381	A2	10	-
BLEKKO	true	true	WH_PROP_BE	0.26	0.377	A1	5	-
BLEKKO	true	true	WH_PROP_BE	0.23	0.334	A2	10	Max
BLEKKO	true	true	WH_PROP_BE	0.22	0.299	A1	10	-
BLEKKO	true	true	WH_PROP_BE	0.21	0.319	A1	5	Sum
BLEKKO	true	true	WH_PROP_BE	0.21	0.305	A1	10	Max
BLEKKO	true	true	WH_PROP_BE	0.20	0.290	A2	5	-
BLEKKO	true	true	WH_PROP_BE	0.19	0.289	A2	5	Max
BLEKKO	true	true	WH_PROP_BE	0.19	0.276	A2	10	Sum
BLEKKO	true	true	WH_PROP_BE	0.16	0.243	A1	5	Max
BLEKKO	true	true	WH_PROP_BE	0.15	0.242	A1	1	Sum
BLEKKO	true	true	WH_PROP_BE	0.15	0.236	A2	1	-
BLEKKO	true	true	WH_PROP_BE	0.14	0.225	A2	1	Sum
BLEKKO	true	true	WH_PROP_BE	0.14	0.220	A1	1	-

Table D.1 – Continued

Search	AQ	Quote	Filter	Acc.	C@1	Sim.	Passages	Norm.
BLEKKO	true	true	WH_PROP_BE	0.14	0.213	A2	5	Sum
BLEKKO	true	true	WH_PROP_BE	0.14	0.203	A1	10	Sum
BLEKKO	true	true	WH_PROP_BE	0.12	0.193	A1	1	Max
BLEKKO	true	true	WH_PROP_BE	0.12	0.193	A2	1	Max

Table D.2: Results for LSA, for Portuguese WWBM corpus.

Search	AQ	Quote	Filter	Acc.	C@1	App.	Passages	Norm.
Bing	true	true	BE	0.29	0.342	A2	5	-
Bing	true	true	BE	0.29	0.342	A2	10	-
Bing	true	true	BE	0.28	0.330	A1	5	Max
Bing	true	true	BE	0.28	0.330	A1	5	Sum
Bing	true	true	BE	0.27	0.319	A1	5	-
Bing	true	true	BE	0.26	0.307	A2	5	Max
Bing	true	true	BE	0.25	0.295	A1	10	-
Bing	true	true	BE	0.22	0.290	A2	1	Max
Bing	true	true	BE	0.21	0.277	A2	1	-
Bing	true	true	BE	0.21	0.277	A2	1	Sum
Bing	true	true	BE	0.21	0.250	A1	10	Sum
Bing	true	true	BE	0.20	0.264	A1	1	Sum
Bing	true	true	BE	0.20	0.238	A2	10	Sum
Bing	true	true	BE	0.20	0.236	A2	5	Sum
Bing	true	true	BE	0.19	0.226	A1	10	Max
Bing	true	true	BE	0.18	0.238	A1	1	-
Bing	true	true	BE	0.18	0.214	A2	10	Max
Bing	true	true	BE	0.15	0.198	A1	1	Max