INSTITUTO SUPERIOR TÉCNICO
Universidade Técnica de Lisboa

The Answer Question in Question Answering Systems

André Filipe Esteves Gonçalves

Dissertação para obtenção do Grau de Mestre em

Engenharia Informática e de Computadores

Júri

Presidente:	Prof. José Manuel Nunes Salvador Tribolet
Orientador:	Prof^a Maria Luísa Torres Ribeiro Marques da Silva Coheur
Vogal:	Prof. Bruno Emanuel da Graça Martins

Novembro 2011

Acknowledgements

I am extremely thankful to my Supervisor, Prof. Luísa Coheur and also my colleague Ana Cristina Mendes (who pratically became my Co-Supervisor) for all their support throughout this journey. I could not have written this thesis without them, truly.

I dedicate this to everyone that has inspired me and motivates me on a daily basis over the past year (or years) – i.e. my close friends, my chosen family. It was worthy. Thank you, you are my heroes.

Resumo

No mundo atual, os Sistemas de Pergunta Resposta tornaram-se uma resposta válida para o problema da explosão de informação da Web, uma vez que conseguem efetivamente compactar a informação que estamos à procura numa única resposta. Para esse efeito, um novo Sistema de Pergunta Resposta foi criado: Just.Ask.

Apesar do Just.Ask já estar completamente operacional e a dar respostas corretas a um certo número de questões, permanecem ainda vários problemas a endereçar. Um deles é a identificação errónea de respostas a nível de extração e clustering, que pode levar a que uma resposta errada seja retornada pelo sistema.

Este trabalho lida com esta problemática, e é composto pelas seguintes tarefas: a) criar corpora para melhor avaliar o sistema e o seu módulo de respostas; b) apresentar os resultados iniciais do Just.Ask, com especial foco para o seu módulo de respostas; c) efetuar um estudo do estado da arte em relações entre respostas e identificação de paráfrases de modo a melhorar a extração das respostas do Just.Ask; d) analizar erros e detectar padrões de erros no mdulo de extração de respostas do sistema; e) apresentar e implementar uma solução para os problemas detectados. Isto incluirá uma normalização/integração das respostas potencialmente corretas, juntamente com a implementação de novas técnicas de ”clustering”; f) avaliação do sistema com as novas adições propostas.

Abstract

In today's world, Question Answering (QA) Systems have become a viable answer to the problem of information explosion over the web, since they effectively compact the information we are looking for into a single answer. For that effect, a new QA system called Just.Ask was designed.

While Just.Ask is already fully operable and giving correct answers to certain questions, there are still several problems that need to be addressed. One of them is the erroneous identification of potential answers and clustering of answers, which can lead to a wrong answer being returned by the system.

This work deals with this problem, and is composed by the following tasks: a) create test corpora to properly evaluate the system and its answers module; b) present the baseline results of Just.Ask, with special attention to its answers module; c) perform a state of the art in relating answers and paraphrase identification in order to improve the extraction of answers of Just.Ask; d) analyze errors and detect error patterns over the answer extraction stage; e) present and implement a solution to the problems detected. This will include normalization/integration of the potential answers as well as implementing new clustering techniques; f) Evaluation of the system with the new features proposed.

Keywords

Just.Ask
Question Answering
Relations between Answers
Paraphrase Identification
Clustering
Normalization
Corpus
Evaluation
Answer Extraction

Index

List of Figures

List of Tables

Acronyms

TREC – Text REtrieval Conference
QA – Question-Answering
POS – Part of Speech

1 Introduction

QA Systems have been with us since the second half of the 20th century. But it was with the world wide web explosion that QA systems truly broke out, leaving restricted domains for global, worldwide ones.

With the exponential growth of information, finding specific information quickly became an issue. Search engines were and still are a valid option, but the amount of the noisy information they generate may not be worth the shot – after all, most of the times we just want a specific, single, 'correct' answer instead of a set of documents deemed relevant by the engine we are using. The task of finding the most 'correct' answer for a given question should be the main purpose of a QA system, after all.

This led us to the development of Just.Ask [39, 38], a QA system with the mission to retrieve (as any other QA system) answers from the web to a given question. Just.Ask combines hand-crafted rules and machine-learning components, and it is composed by the same modules that define a general QA architecture: Question Interpretation, Passage Retrieval and Answer Extraction. Unfortunately, the Answer Extraction module is not as efficient as we would like it to be.

QA systems usually base the process of finding an answer using the principle of redundancy [22], i.e., they consider the 'right' answer as the one that appears more often, in more documents. Since we are working with the web as our corpus, we are dealing with the risk of several variations of the same answer being perceived as different answers, when in fact they are only one, and should be counted as such. To solve this problem, we need ways to connect all of these variations.

In resume, the goals of this work are:

- To perform a state of the art on answers' relations and paraphrase identification, taking into special account the approaches that can be more easily adopted by a QA system such as Just.Ask;
- improve the Answer Extraction module by finding/implementing a proper methodology/tools to correctly identify relations of equivalence and inclusion between answers through a relations module of sorts;
- test new clustering measures;
- create corpora and software tools to evaluate our work.

This document is structured in the following way: Section 2 describes specific related work in the subject of relations between answers and paraphrase identification, Section 3 details our point of departure – Just.Ask system – and the tasks planned to improve its Answer Extraction module along with the baseline results prior to this work, Section 4 describes the main work done in relating anwers and respective results, Section 5 details the work done for the Clustering stage and the results achieved, and finally, on Section 6, we present the main conclusions and future work.

2 Related Work

As previously stated, the focus of this work is to find a better formula to catch relations of equivalence, from notational variances to paraphrases. First, we identify the relations between answers that will be the target of our study. And since paraphrases are such a particular type of equivalences (and yet with the potential to encompass all types of equivalences, if we extend the concept enough), we then talk about the different techniques to detect paraphrases described in the literature.

2.1 Relating Answers

As we said in Section 1, QA systems usually base the process of recovering answers on the principle of redundancy, i.e., every information item is likely to have been stated multiple times, in multiple ways, in multiple documents [22]. But this strategy only works if it has a methodology of relating answers behind, and that is still a major task in QA.

Our main goal here is to learn how a pair of answers are related so that our system can retrieve a more correct final answer, based on frequency count and the number and type of relationships between pairs present in the relevant passages.

From the terminology first proposed by Webber and his team [44] and later described by Moriceau [29], we can define four different types of relationships between answers:

Equivalence – if answers are consistent and entail mutually: whether notational variations ('John F. Kennedy' and 'John Kennedy' or '24 October 1993' and 'October 23rd 1993') or paraphrases (from synonyms and rephrases, such as 'He was shot and died' vs. 'He was murdered' to inferences, such as 'Judy Garland was Liza Minelli's mother' and 'Liza Minelli was Judy Garland's sister'.)

Inclusion – if answers are consistent and differ in specificity, one entailing the other. They can be of hypernymy (for the question 'What kind of animal is Simba?', can be answered with 'lion' or 'mammal') or meronymy (for the question 'Where is the famous La Scala opera house', 'Milan', 'Italy' and 'Europe' are all possible answers).

Alternative – if answers are not entailing at all; they constitute valid complementary answers, whether from the same entity ('molten rock' and 'predominantly 8 elements (Si, O, Al, Mg, Na, K, Ca, Fe)' from the question 'What does magma consist of'), or representing different entities (all the answers to the question 'Name a German philosopher').

Contradictory – if answers are inconsistent and their conjunction is invalid. For instance, the answers '1430km' and '1,609 km' for the question 'How long is the Okavango River' are contradictory.

The two measures that will be mentioned in Section 3.1 to cluster answers are still insufficient for a good clustering at Just.Ask. For some answers, like

'George Bush' and 'George W. Bush', we may need semantic information, since we may be dealing with two different individuals.

2.2 Paraphrase Identification

According to WordReference.com [1] a paraphrase can be defined as:

– (noun) rewording for the purpose of clarification

– (verb) express the same message in different words

There are other multiple definitions (other paraphrases of the definition of the word 'paraphrase'). In the context of our work, we define paraphrases as expressing the same idea in two different text fragments, in a symmetric relationship in which one entails the other – meaning potential addition of information, with only one fragment entailing the other, should be discarded as such.

Here is an example of two sentences which should be deemed as paraphrases taken from the Microsoft Research Paraphrase Corpus [2], from now on mentioned as the MSR Corpus:

> Sentence 1: Amrozi accused his brother, whom he called 'the witness', of deliberately distorting his evidence.
> Sentence 2: Referring to him as only 'the witness', Amrozi accused his brother of deliberately distorting his evidence.

In the following pages, we will present an overview of some of approaches and techniques to paraphrase identification related to natural language, specially after the release of the MSR Corpus.

2.2.1 SimFinder – A methodology based on linguistic analysis of comparable corpora

We can view two comparable texts as collections of several potential paraphrases. SimFinder [9] was a tool made for summarization from a team led by Vasileios Hatzivassiloglou, that allowed to cluster highly related textual units. In 1999, they defined a more restricted concept of similarity in order to reduce ambiguities: two text units are similar if 'they share the same focus on a common concept, actor, object, or action'. In addition to that, the common object/actor must either be the subject of the same description or must perform (or be subjected to) the same action. The linguistic features implemented can be grouped into two types:

[1]http://www.wordreference.com/definition/paraphrase

[2]The Microsoft Research Paraphrase Corpus (MSR Paraphrase Corpus) is a file composed of 5800 pairs of sentences which have been extracted from news sources over the web, with manual (human) annotations indicating whether a pair is a paraphrase or not. It is used in a considerable amount of work in Paraphrase Identification.

– Primitive Features – encompassing several ways to define a match, such as word co-occurrence (sharing a single word in common), matching noun phrases, shared proper nouns, WordNet [28] synonyms, and matching verbs with the same semantic class. These matches not only consider identical words, but also words that match on their stem – the most basic form of a word, from which affixes are attached.

– Composite Features – combination of pairs of primitive features to improve clustering. Analyzing if two joint primitives occur in the same order, within a certain distance, or matching joint primitives based on the primitives' type – for instance, matching a noun phrase and a verb together.

The system showed better results than TD-IDF [37], the classic information retrieval method, already in its first version. Some improvements were made later on, with a larger team that included Regina Barzilay, a very prominent researcher in the field of Paraphrase Identification, namely the introduction of a quantitative evaluation measure.

2.2.2 Methodologies purely based on lexical similarity

In 2003, Barzilay and Lee [3] used the simple N-gram overlap measure to produce clusters of paraphrases. The N-gram overlap is used to count how many N-grams overlap in two sentences (usually with N $\leq$ 4). In other words: how many different sequences of words we can find between two sentences. The following formula gives us the measure of the Word Simple N-gram Overlap:

$$NGsim(S1, S2) = \frac{1}{N} * \sum_{n=1}^{N} \frac{Count\_match(N-gram)}{Count(N-gram)} \tag{1}$$

in which Count_match(N-gram) counts how many N-grams overlap for a given sentence pair, and Count(N-gram) represents the maximum number of N-grams in the shorter sentence.

A year later, Dolan and his team [7] used Levenshtein Distance [19] to extract sentence-level paraphrases from parallel news sources, and compared it with an heuristic strategy that paired initial sentences from different news stories. Levenshtein metric calculates the minimum number of editions to transform one string into another. Results showed that Levenshtein data was cleaner and more easily aligned, but the Alignment Error Rate (AER), an objective scoring function to measure how well an algorithm can align words in the corpus combining precision and recall metrics described in [31], showed that the two strategies performed similarly, each one with strengths and weaknesses. A string edit distance function like Levenshtein achieved high precision, but involved inadequate assumptions about the data, while the heuristic allowed to extract more rich but also less parallel data.

In 2007, Cordeiro and his team [5] analyzed previous metrics, namely Levenshtein Distance and Word N-Gram overlap family of similarity measures [3, 32], and concluded that these metrics were not enough to identify paraphrases, leading to wrong judgements and low accuracy scores. The Levenshtein distance and an overlap distance measuring how many words overlap (see Section 3.1.3) are ironically the two metrics that the actual answer extraction module at Just.Ask possesses at the moment, so this is a good starting point to explore some other options. As an alternative to these metrics, a new category called *LogSim* was proposed by Cordeiro's team. The idea is, for a pair of two input sentences, to count exclusive lexical links between them. For instance, in these two sentences:

S1: Mary had a very nice dinner for her birthday.
S2: It was a successful dinner party for Mary on her birthday.

there are four links ('dinner', 'Mary', 'her', 'birthday'), i.e., four common words between S1 and S2. The function in use is:

$$L(x, \lambda) = -\log 2(\lambda/x) \tag{2}$$

where x is the number of words of the longest sentence and λ is the number of links between S1 and S2 (with the logarithm acting as a slow penalty for too similar sentences that are way too similar). The complete formula is written below:

$$logSim(S1, S2) = \begin{cases} L(x, \lambda) & if L(x, \lambda) < 1.0 \\ e^{-k*L(x,\lambda)} & otherwise \end{cases} \tag{3}$$

This ensures that the similarity only variates between 0 and 1. The idea behind the second branch is to penalize pairs with great dissimilarities boosted by a constant k (used as 2.7 in their experiments), as $\frac{\lambda}{x}$ tends to 0 and therefore $L(x, \lambda)$ tends to infinity. However, having a metric depending only on x, and not y, i.e., the number of words in the shortest sentence, is not a perfect measure – after all, they are not completely independent, since $\lambda \leq$ y. Therefore, an expanded version of *LogSim* was also developed and tested by the same team, now accepting three parameters: λ, x and y.

$L(x, y, z)$ is computed using weighting factors α and β (β being (1-α)) applied to L(x, λ) and L(y, λ), respectively. They used three self-constructed corpus using unions of the MSR Corpus, the Knight and Marcu Corpus [3], and extra negative pairs (of non-paraphrases), to compensate for the lack of negative pairs offered by the two corpus. When compared to the Levenshtein and N-gram metrics, *LogSim* performed substantially better when the MSR Paraphrase Corpus was present in the corpus test.

[3]This was a corpus used by Kevin Knight and Daniel Marcu, produced manually from pairs of texts and respective summaries, containing 1087 similar sentence pairs, with one sentence being a compressed version of another.

2.2.3 Canonicalization as complement to lexical features

Zhang and Patrick [47] offer another sentence-level solution, converting the pairs of sentences into canonical forms at the lexical and syntactic level – 'generic' versions of the original text – and then applying lexical matching features such as Edit Distance and N-gram overlap. This canonicalization involves particularly number entities (dates, times, monetary values, percentages), passive to active voice transformation (through dependency trees, using Minipar [21]), and future tense sentences (mapping structures such as 'plans to' or 'expects to do' into the word 'will'). They also claimed that the MSR Paraphrase Corpus is limited, by counting the number of the nominalizations (i.e. producing a noun from another part of the speech, like 'combination' from 'combine') through a semi-automatic method and concluding that the MSR has many nominalizations and not enough variety – which discards a certain group of paraphrases – the ones that use (many) different words to express the same meaning. This limitation can also explain why this method performs as well as using a baseline approach. Moreover, the baseline approach with simple lexical matching features led to better results than other more 'sophisticated' methodologies, which again should support the evidence of the lack of lexical variety in the corpus used. Just.Ask already has a normalizer for answers of type NUMERIC:COUNT and NUMERIC:DATE, so the kind of equivalences involving Numeric Entities that can be considered answers for certain types of questions are already covered. Passive to Active Voice and Future Tense transformations can however matter to more particular answers. Considering for instance the question 'How did President John F. Kennedy die?', the answers 'He was shot' and 'Someone shot him' are paraphrases in the passive and active voice respectively.

2.2.4 Use of context to extract similar phrases

While dealing with the task of sentence alignment in monolingual copora in the field of text-to-text generation, Barzilay and Elhadad [2] raised a very pertinent question. Two sentences sharing most of their words are likely to be paraphrases. But these are the so-called 'easy' paraphrases. There are many paraphrases that convey the same information but that share little lexical resemblance. And for that, the usual lexical similarity measures mentioned above should not be enough by themselves. They proposed a two-phase process, first using learning rules to match pairs of paragraphs based on similar topic structure (with two paragraphs being grouped because they convey the same type of information, such as weather reports for instance), and then refining the match through local alignement. Not only the easy 'paraphrases' were identified, but also some of the tricky ones, by combining lexical similarity (a simple cosine measure) with dynamic programming. Considering two sentences S1 and S2, their local similarity is given by:

$$sim(S1, S2) = cos(S1, S2) - mismatch\_penalty \tag{4}$$

with the mismatch penalty being a decimal value used to penalize pairs with very low similarity measure. The weight of the optimal alignment between two pairs is given through the following formula of dynamic programming – a way of breaking down a complex program into a series of simpler subproblems:

$$s(S1, S2) = max \begin{cases} s(S1, S2-1) - skip\_penalty \\ s(S1-1, S2) - skip\_penalty \\ s(S1-1, S2-1) + sim(S1, S2) \\ s(S1-1, S2-2) + sim(S1, S2) + sim(S1, S2-1) \\ s(S1-2, S2-1) + sim(S1, S2) + sim(S1-1, S2) \\ s(S1-2, S2-2) + sim(S1, S2-1) + sim(S1-1, S2) \end{cases} \tag{5}$$

with the skip penalty preventing only sentence pairs with high similarity from getting aligned.

Results were interesting. A 'weak' similarity measure combined with contextual information managed to outperform more sophisticated methods such as SimFinder, using two collections from Encyclopedia Britannica and Britannica Elementary. Context matters.

This whole idea should be interesting to explore in a future line of investigation within Just.Ask where we study the answer's context, i.e., the segments where the answer has been extracted. This work has also been included because it warns us for the several different types of paraphrases/equivalences that can exist in a corpus (more lexical-based or using completely different words).

2.2.5 Use of dissimilarities and semantics to provide better similarity measures

In 2006, Qiu and colleagues [33] have proposed a two-phased framework using dissimilarity between sentences. In the first phase, similarities between tuple pairs (more than a single word) were computed using a similarity metric that takes into account both syntactic structure and tuple content (derived from a thesaurus). Then, the remaining tuples that found no relevant match (unpaired tuples), which represent the extra bits of information than one sentence contains and the other does not, are parsed through a dissimilarity classifier, and thus to determine if we can deem the two sentences as paraphrases.

Mihalcea and her team [27] decided that a semantical approach was essential to identify paraphrases with better accuracy. By measuring semantic similarity between two segments, combining corpus-based and knowledge-based measures, they were able to get much better results than a standard vector-based model using cosine similarity.

They took into account not only the similarity but also the 'specificity' between two words, giving more weight to a matching between two specific words (like two tonalities of the same color, for instance) than two generic concepts. This specificity is determined using inverse document frequency (IDF) [37], given by the total number of documents in corpus divided by the total number of documents including the word. For the corpus, they used British National Corpus [4], a collection of 100 million words, that gathers both spoken-language and written samples.

The similarity of two text segments T1 and T2 (sim(T1,T2)) is given by the following formula:

$$\frac{1}{2}\left(\frac{\sum_{w\in\{T1\}}(maxsim(w,T2).idf(w))}{\sum_{w\in\{T1\}} idf(w)} + \frac{\sum_{w\in\{T2\}}(maxsim(w,T1).idf(w))}{\sum_{w\in\{T2\}} idf(w)}\right) \tag{6}$$

For each word w in the segment T1, the goal is to identify the word that matches the most in T2, given by maxsim (w, T2), and then apply the specificity via IDF, make a sum of these two methods, and normalize it with the length of the text segment in question. The same process is applied to T2, and in the end we generate an average between T1 to T2 and vice versa. They offer us eight different similarity measures to calculate maxsim: two corpus-based (Pointwise Mutual Information [42] and Latent Semantic Analysis – LSA [17]) and six knowledge-based (Leacock and Chodorow–lch [18], Lesk [1], Wu & Palmer–wup [46], Resnik [34], Lin [22], and Jiang & Conrath–jcn [14]). Below, we offer a brief description of each of these metrics:

– Pointwise Mutual Information (PMI-IR) – first suggested by Turney [42] as a measure of semantic similarity of words, is based on the word co-occurrence using counts collected over very large corpora. Given two words, their PMI-IR score indicates a degree of dependence between them;

– Latent Semantic Analysis (LSA) – uses a technique in algebra called Singular Value Decomposition (SVD), which in this case decomposes the term-document matrix into three matrixes (reducing the linear system into multidimensional vectors of semantic space). A term-document matrix is composed of passages as rows and words as columns, with the cells containing the number of times a certain word was used in a certain passage.

– Lesk – measures the number of overlaps (shared words) in the definitions (glosses) of two concepts and concepts directly related through relations of hypernymy and meronymy;

– Lch – computes the similarity between two nodes by finding the path length between two nodes in the $is-a$ hierarchy. An $is-a$ hierarchy defines specializations (if we travel down the tree) and generalizations between concepts (if we go up);

– Wup – computes the similarity between two nodes by calculating the path length from the least common subsumer (LCS) of the nodes, i.e. the most

[4]http://www.natcorp.ox.ac.uk/

specific node two nodes share in common as an ancestor. If one node is 'cat' and the other is 'dog' the LCS would be 'mammal';

– Resnik – takes the LCS of two concepts and computes an estimation of how 'informative' the concept really is (its Information Content). A concept that occurs frequently will have low Information Content, while a concept that is quite rare will have a high Information Content;

– Lin – takes the Resnik measure and normalizes it, by using also the Information Content of the two nodes received as input;

– Jcn – uses the same information as Lin, but combined in a different way.

Kozareva, Vazquez and Montoyo [16] also recognize the importance of having semantic knowledge in Textual Entailment Recognition (RTE) methods, i.e., methods to decide if one text can be entailed in another and vice versa – in other words, paraphrase identification. Moreover, they present a new model in order to detect semantic relations between word pairs from different syntactic categories. Comparisons should not be only limited between two verbs, two nouns or two adjectives, but also between a verb and a noun ('assassinated' *vs.* 'assassination'), or an adjective and a noun ('happy' *vs.* 'happiness'), for instance. They measure similarity, using variations of features such as:

– Sentence Expansion – expanding nouns, verbs, adjectives and adverbs to their synonyms, antonyms and verbs that precede/entail the verbs in question. Word sense disambiguation is used to reduce the potential noise of such expansion;

– Latent Semantic Analysis (LSA) – by constructing a term-domain matrix instead of a term-document one, in which the domain is extracted from the WordNet domain resource;

– Cosine measure – by using Relevant Domains instead of word frequency.

Fernando and Stevenson [8] also considered that a purely lexical similarity approach such as the one conducted by Zhang and Patrick was quite limited, since it failed to detect words with the same meaning, and therefore proposed a semantic similarity approach based on similarity information derived from WordNet, and using the concept of matrix similarity first developed by Stevenson and Greenwood in 2005 [41].

Each input sentence is mapped into a binary vector, with the elements equal to 1 if a word is present and 0 otherwise. For a pair of sentences S1 and S2, we have therefore two vectors: $\vec{a}$ and $\vec{b}$, and the similarity is calculated using the following formula:

$$sim(\vec{a},\vec{b}) = \frac{\vec{a}.W.\vec{b}}{\parallel \vec{a} \parallel . \parallel \vec{b} \parallel} \tag{7}$$

with W being a semantic similarity matrix containing the information about the similarity level of a word pair – in other words, a number between 0 and 1 for each Wij, representing the similarity level of two words, one from each sentence. Six different WordNet metrics were used (the already mentioned Lch, Lesk, Wup, Resnik, Lin and Jcn), five of them with an accuracy score higher than either Mihalcea, Qiu or Zhang and five with also a better precision rating than those three metrics.

More recently, Lintean and Rus [23] identify paraphrases of sentences using weighted dependences and word semantics, taking into account both the similarity and the dissimilarity between two sentences (S1 and S2), and then using them to compute a final paraphrase score as the ratio of the similarity and dissimilarity scores:

$$Paraphrase(S1; S2) = Sim(S1, S2)/Diss(S1, S2). \tag{8}$$

If this score is above a certain threshold T, the sentences are deemed paraphrases; otherwise, they are not considered paraphrases. The threshold is obtained by optimizing the performance of the approach on training data. They used word-to-word similarity metrics based on statistical information from WordNet which can potentially identify semantically related words, even if they are not exactly synonyms (they give the example of the words 'say' and 'insist'). These similarity metrics are obtained through weighted dependency (syntactic) relationships [10] built between two words in a sentence: a head and a modifier. The weighting is done considering two factors: the type/label of the dependency, and the depth of a dependency in the dependency tree. Minipar [21] and the Stanford parser [4] were used to extract dependency information. To compute the scores, they first map the input sentences into sets of dependencies, detect the common and non-common dependencies between the two and then they calculate the Sim(S1,S2) and Diss(S1,S2) parameters. If two sentences refer to the same action but one is substantially longer than the other (i.e. it adds much more information), then there is no two-way entailment, and therefore the two sentences are non-paraphrases. Lintean and Rus decided to add a constant value (the value 14) when the set of unpaired dependencies of the longer sentence has more than 6 extra dependencies than the set of unpaired dependencies of the shorter one. The duo managed to improve Qiu's approach by using word semantics to compute similarities, and by using dependency types and the position in the dependency tree to weight dependencies instead of simply using unlabeled and unweighted relations.

2.3 Overview

Thanks to the work initiated by Webber et al., we can now define four different types of relations between candidate answers: equivalence, inclusion, contradiction and alternative. Our focus will be on detecting relations of equivalence, expanding the two metrics already available in Just.Ask (Levenshtein distance and overlap distance).

We saw many different types of techniques over the last pages, and many more have been developed over the last ten years, presenting variations of the ideas which have been described, such as building dependency trees or using semantical metrics dealing with the notions such as synonymy, antonymy and hyponymy. For the purpose of synthesis, the work here represents what we found with the most potential of being successfully tested on our system.

There are two basic lines of work here: the use of lexical-based features, such as the already implemented Levenshtein distance, and the use of semantic information (from WordNet). From these, there were several interesting features attached such as: the transformation of units into canonical forms, the potential use of context surrounding the answers we extract as candidates, dependency trees, similarity matrixes, and the introduction of a dissimilarity score in the formula of paraphrase identification.

For all of these techniques, we also saw different types of corpus for evaluation. The most prominent and most widely available is MSR Paraphrase Corpus, which was criticized by Zhang and Patrick for being composed of too many lexically similar sentences. Other corpus used was Knight and Marcu corpus (KMC), composed of 1087 sentence pairs. Since these two corpus lack negative pairs, i.e. pairs of non-paraphrases, Cordeiro et al. decided to create three new corpus, using a mixture of these two with negative pairs in order to balance the number of positive pairs and negative ones. Finally, British National Corpus (BNC) was used for the semantic approach of Mihalcea et al.

3 Just.Ask architecture and work points

The work of this thesis will be done on Just.Ask [39], a QA system developed at INESC-ID[5] that combines rule-based components with machine learning-based ones. Over this chapter, we will describe the architecture of this system, focusing on the answers module, and show the results already attained by the system in general and by the Answer Extraction module. We then detail what we plan to do to improve this module.

3.1 Just.Ask

This section describes what has been done so far with Just.Ask. As said before, the system's architecture follows the standard QA architecture with three stages: Question Interpretation, Passage Retrieval and Answer Extraction. The picture below shows the basic architecture of Just.Ask. We will work over the Answer Extraction module, dealing with relationships between answers and paraphrase identification.

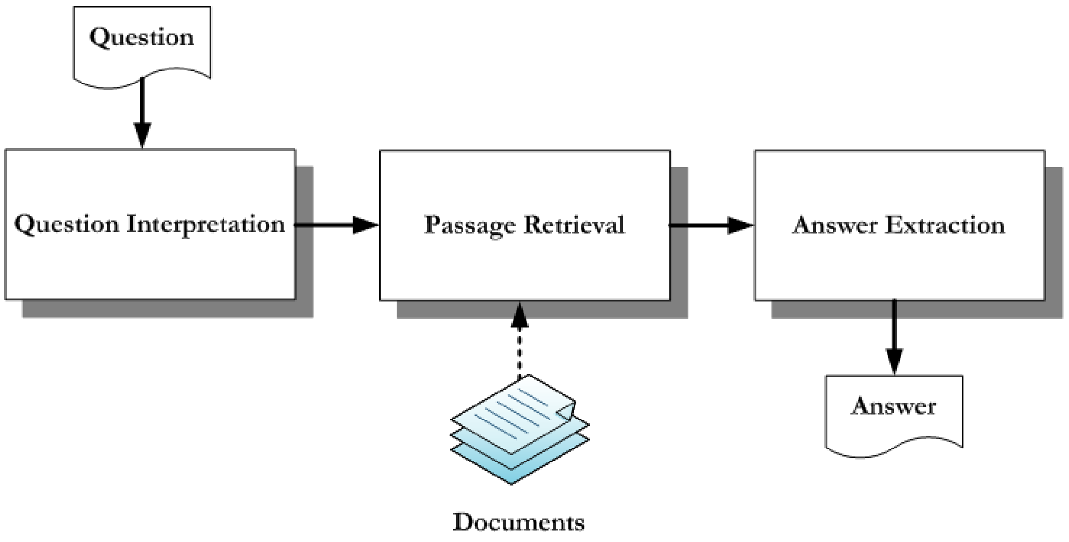


Figure 1: Just.Ask architecture, mirroring the architecture of a general QA system

3.1.1 Question Interpretation

This module receives as input a natural language question and outputs an interpreted question. In this stage, the idea is to gather important information

[5]http://www.inesc-id.pt/

about a question (question analysis) and classify it under a category that should represent the type of answer we look for (question classification).

Question analysis includes tasks such as transforming the question into a set of tokens, part-of-speech tagging, and identification of a questions headword, i.e. a word in the given question that represents the information we are looking for. Question Classification is also a subtask in this module. In Just.Ask, Question Classification uses Li and Roth's [20] taxonomy, with 6 main categories (Abbreviation, Description, Entity, Human, Location and Numeric) plus 50 more detailed classes. The framework allows the usage of different classification techniques: hand-built rules and machine-learning. Just.Ask already attains state of the art results in question classification [39].

3.1.2 Passage Retrieval

The passage retrieval module receives as input the interpreted question from the preceding module. The output is a set of relevant passages, i.e., extracts from a search corpus that contain key terms of the question, and are therefore most likely to contain a valid answer. The module is composed by a group of query formulators and searchers. The job here is to associate query formulators and question categories to search types. Factoid-type questions, such as 'When was Mozart born', use Google or Yahoo search engines, while definition questions like 'What is a parrot' use Wikipedia.

As for query formulation, Just.Ask employs multiple strategies according to the question category. The most simple is keyword-based strategy, applied for factoid-type questions, which consists in generating a query containing all the words of a given question. Another strategy is based on lexico-syntatic patterns, to learn how to rewrite a question in order to get us closer to the actual answer. Wikipedia and DBpedia are used in a combined strategy using Wikipedia's search facilities to locate the title of the article where the answer might be found, and DBpedia to retrieve the abstract (the first paragraph) of the article, which is returned as the answer.

3.1.3 Answer Extraction

The answer extraction module receives an interpreted question and a set of relevant passages as input and outputs the final answer. There are two main stages here: Candidate Answer Extraction and Answer Selection. The first deals with extracting the candidate answers, and the latter with normalization, clustering and filtering the candidate answers to give us a correct final answer.

Just.Ask uses regular expressions (for Numeric type questions), named entities (able to recognize four entity types: Person, Location, Organization, and Miscellaneous), WordNet based recognizers (hyponymy relations) and Gazeteers (for the categories Location:Country and Location:City) to extract plausible answers from relevant passages. Currently, only answers of type Numeric:Count and Numeric:Date are normalized. Normalization is important to diminish the answers variation, converting equivalent answers to a canonical representation. For example, the dates 'March 21, 1961' and '21 March 1961' will be transformed into 'D21 M3 Y1961'.

Similar instances are then grouped into a set of clusters, and for that purpose, distance metrics are used, namely overlap distance and Levenshtein [19] distance. The overlap distance is defined as:

$$Overlap(X,Y) = 1 - |X \bigcap Y|/(min(|X|,|Y|), \tag{9}$$

where $|X \bigcap Y|$ is the number of tokens shared by candidate answers X and Y and $min(|X|,|Y|)$ the size of the smallest candidate answer being compared. This formula returns 0.0 for candidate answers that are either equal or contained in the other. As it was mentioned in Section 3.1.3, the Levenshtein distance computes the minimum number of operations in order to transform one string to another. These two distance metrics are used with a standard single-link clustering algorithm, in which the distance between two clusters is considered to be the minimum of the distances between any member of the clusters. Initially, every candidate answer has a cluster of its own, and then, for each step, the two closest clusters are merged if they are under a certain threshold distance. The algorithm halts when the distances of the remaining clusters are higher than the threshold. Each cluster has a representative – the clusters longest answer, and a score is assigned to the cluster, which is simply the sum of the scores of each candidate answer. With the exception of candidate answers extracted using regular expressions, the score of a candidate answer is always 1.0, meaning the clusters score will in many cases be directly influenced by its length.

3.2 Experimental Setup

Over this section, we will describe the corpora we used to evaluate our system.

3.2.1 Evaluation measures

To evaluate the results of Just.Ask, the following measures suited to evaluate QA systems [43] are used:

Precision:

$$Precision = \frac{\#Correctly\ judged\ items}{\#Relevant\ (non-null)\ items\ retrieved} \tag{10}$$

Recall:

$$Recall = \frac{\#Relevant\ (non-null)\ items\ retrieved}{\#Items\ in\ test\ corpus} \tag{11}$$

F-measure (F1):

$$F = 2 * \frac{precision * recall}{precision + recall} \tag{12}$$

Accuracy:

$$Accuracy = \frac{\#Correctly\ judged\ items}{\#Items\ in\ test\ corpus} \tag{13}$$

Mean Reciprocal Rank (MRR): used to evaluate systems that produce a list of possible responses to a query, ordered by probability of correctness. The reciprocal rank of a query response is the multiplicative inverse of the rank of the first correct answer. The mean reciprocal rank is the average of the reciprocal ranks of results for a sample of queries Q in a test corpus. For a test collection of N queries, the mean reciprocal rank is given by:

$$MRR = \frac{1}{N} \sum_{i=1}^{N} \frac{1}{rank_i} \,. \tag{14}$$

3.2.2 Multisix corpus

To know for sure that a certain system works, we have to evaluate it. In the case of QA systems, and with Just.Ask in particular, we evaluate it giving it questions as input, and then we see if the answers the system outputs match the ones we found for ourselves.

We began to use the Multisix corpus [24]. It is a collection of 200 English questions covering many distinct domains composed for the cross-language tasks at CLEF QA-2003. The corpus has also the correct answer for each question, searched in the Los Angeles Times over 1994. We call references to the set of correct answers for each of the 200 questions. A manual validation of the corpus was made, adding to the references many variations of the answers found over the web sources (such as adding 'Imola, Italy', 'Imola' and 'Italy' to the question 'Where did Ayrton Senna have the accident that caused his death'). In the context of this thesis, besides this manual validation, an application was developed to create the test corpus of the references in a more efficient manner.

The 200 questions were also manually classified, according to Li and Roth's question type taxonomy. A few answers were updated, since the question varied in time (one example is the question 'Who is the coach of Indiana'), or even eliminated (10 of the 200, to be more accurate), since they no longer made sense (like asking who is the vice-president of a company that no longer exists, or how old is a person that has already died). And since the Multisix corpus lacked on DESCRIPTION:DESCRIPTION and HUMAN:DEFINITION questions, 12 questions were gathered and used from the Multieight corpus [25].

For each of the 200 questions, we have a set of references – possible correct answers – initially being updated manually from the first 64 passages that we found over at each of the two main search engines (Google and Yahoo) – which we call 'snippets'. This process is quite exhaustive itself. The corpus file with all the results that the Google/Yahoo query produces is huge (a possible maximum of 64*200 entries!) and to review it takes a considerable amount of time and patience. Moreover, we do not need all the information in it to add a reference. Taking, for instance, this entry:

1 Question=What is the capital of Somalia

{ URL= http://en.wikipedia.org/wiki/Mogadishu Title=Mogadishu - Wikipedia, the free encyclopedia Content=Mogadishu is the largest city in Somalia and the nation's capital. Located in the coastal Benadir region on the Indian Ocean, the city has served as an ... Rank=1 Structured=false}

The attributes 'URL', 'Title', 'Rank' and 'Structured' are irrelevant for the mere process of adding a single (or multiple) references. The only attribute that really matters to us here is 'Content' – our snippet (or relevant passage).

An application was made in order to automate some of the process involved in adding references to a given question – creating the corpus containing every reference captured for all the questions. This application receives two files as arguments: one with a list of questions, and their respective references; and the corpus file, containing the corpus with a stipulated maximum of 64 snippets per question. The script runs the list of questions, one by one, and for each one of them presents all the snippets, one by one, removing 'unnecessary' content like 'URL' or 'Title', requesting an input from the command line.

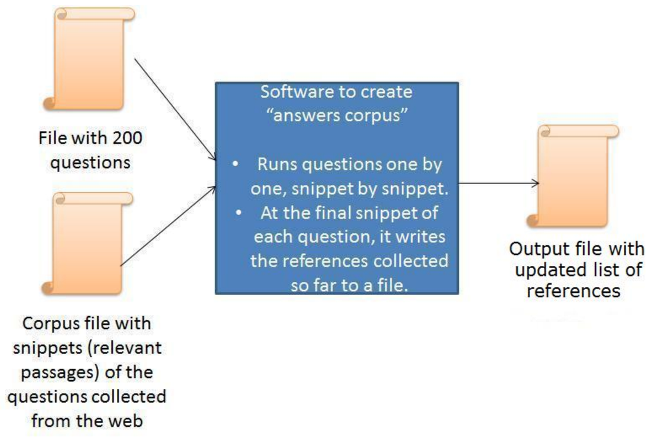
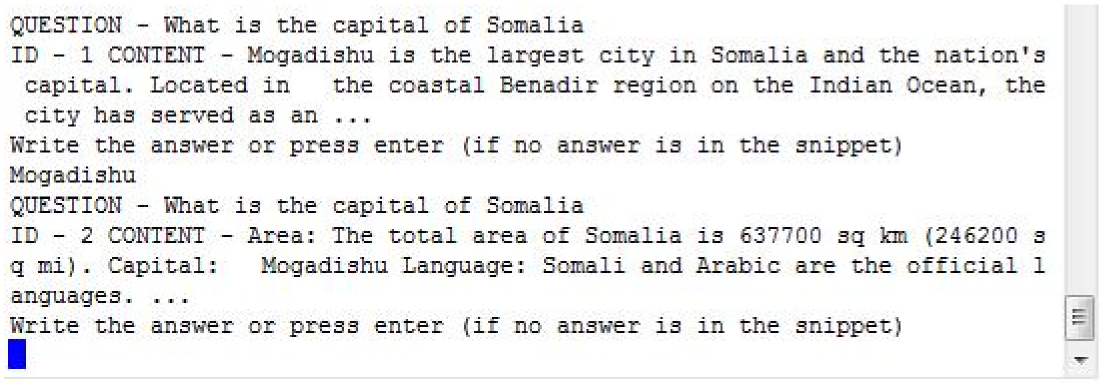

File with 200 questions
Corpus file with snippets (relevant passages) of the questions collected from the web
Software to create "answers corpus"
• Runs questions one by one, snippet by snippet.
• At the final snippet of each question, it writes the references collected so far to a file.
Output file with updated list of references

Figure 2: How the software to create the 'answers corpus' works.

This input is the reference one picks from a given snippet. So, if the content is: 'Mogadishu is the largest capital of Somalia... ', if we insert 'Mogadishu', the system recognizes it as a true reference – since it is included in the content section, and adds it to the 'references list', which is nothing more than an array of arrays. Of course this means any word present in the content is a viable reference, and therefore we can write any word in the content which has no relation whatsoever to the actual answer. But it helps one finding and writing down correct answers, instead of a completely manual approach, like we did before writing the application. In this actual version, the script prints the status of the reference list to an output file, as it is being filled, question by question, by the final snippet of each question. Figure 2 illustrates the process, while Figure 3 presents a snap shot of the application running. This software has also the advantage of potentially construct new corpora in the future.



```
QUESTION - What is the capital of Somalia
ID - 1 CONTENT - Mogadishu is the largest city in Somalia and the nation's
 capital. Located in   the coastal Benadir region on the Indian Ocean, the
 city has served as an ...
Write the answer or press enter (if no answer is in the snippet)
Mogadishu
QUESTION - What is the capital of Somalia
ID - 2 CONTENT - Area: The total area of Somalia is 637700 sq km (246200 s
q mi). Capital:   Mogadishu Language: Somali and Arabic are the official l
anguages. ...
Write the answer or press enter (if no answer is in the snippet)
```

Figure 3: Execution of the application to create answers corpora in the command line

3.2.3 TREC corpus

In an effort to use a more extended and 'static' corpus, we decided to use additional corpora to test Just.Ask. Therefore, in addition to Google and Yahoo passages, we now have also 20 years of New York Times editions, from 1987 to 2007; and in addition to the 200 test questions, we found a different set at TREC which provided a set of 1768 questions, from which we filtered to extract the questions we knew the New York Times had a correct answer. 1134 questions remained after this filtering stage.

TREC questions

The TREC data is composed by a set of questions and also by a set of correct/incorrect judgements from different sources according to each question (such as the New York Times online editions) – answers that TREC had judged as correct or incorrect. We therefore know if a source manages to answer correctly to a question and what is the 'correct' answer it gives. A small application was created using the file with all the 1768 questions and the file containing all the correct judgements as input parameters, in order to filter only the questions we know the New York Times managed to retrieve a correct answer.

After experimenting with the 1134 questions, we noticed that many of these questions are actually interconnected (some even losing an original reference after this filtering step!), and therefore losing precious entity information by using pronouns for example, which could explain in part the underwhelming results we were obtaining with them. Questions such as 'When did she die?' or 'Who was his father?' remain clearly unanswered due to this lost reference. We simply do not know for now who 'he' or 'she' is. It makes us question: "How much can we improve something that has no clear references, because the question itself is referencing the previous one, and so forth?".

The first and most simple solution was to filter the questions that contained unknown third-party references – pronouns such as 'he', 'she', 'it', 'her', 'him', 'hers' or 'his', plus a couple of questions we easily detected as driving the system to clear and obvious dead ends such as 'How many are there now?'. A total of 232 questions without these features were collected, and then tested.

But there were still much less obvious cases present. Here is a blatant example of how we were still over the same problem:

> question 1129. Q:Where was the festival held?
> question 1130. Q:What is the name of the artistic director of the festival?
> question 1131. Q:When was the festival held?
> question 1132. Q:Which actress appeared in two films shown at the festival?
> question 1133. Q:Which film won the Dramatic Screen Play Award at the festival?
> question 1134. Q:Which film won three awards at the festival?

in this particular case, the original reference 'Sundance Festival' is actually lost, along with vital historic context (which Sundance festival? Is it the 21st? the 25th?). Another blatant example is the use of other complex anaphors, such as 'the company' instead of simply 'it' in the question.

We knew that this corpus was grouped by topics, and therefore was composed by a set of questions for each topic. By noticing that each of these questions from the TREC corpus possessed a topic information that many times was not explicit within the question itself, the best possible solution should be to somehow pass this information to the query at the Passage Retrieval module. By manually inserting this topic information into the questions from TREC 2004 to 2007 (the 2003 questions were independent from each other) in the best manner, and by noticing a few mistakes in the application to filter the questions that had a correct answer from New York Times, we reached a total of 1309 questions. However, due to a length barrier at Just.Ask, we reduced it to 1210 questions.

New York Times corpus

20 years of New York Times editions were indexed using Lucene Search as tool. Instead of sentences or paragraphs as relevant passages, as it happpened with Google and Yahoo search methods, we started to return the whole article as a passage. Results were far from being as famous as with the Google corpus.

The problem here might lie in a clear lack of redundancy – candidate answers not appearing as often in several variations as in Google.

We also questioned if the big problem is within the passage retrieval phase, since we are now returning a whole document, instead of a single paragraph/phrase as we were with the Google corpus. Imagine the system retrieving a document on the question 'What is the capital of Iraq?' that happens to be a political text mentioning the conflict between the country and the United States, and ends up mentioning Washington more times over the document than any Iraq city including the country's capital...

With this in mind, we tried another approach in which the indexed passages would not be the whole documents, but only the paragraph that contains the keywords we send in the query. This change improved the results, but unfortunately not as much as we could hope for, as we will see over the next section.

3.3 Baseline

3.3.1 Results with Multisix Corpus

Results for the overall system before this work are described over Table 1. Precision@1 indicates the precision results for the top answer returned (out of the three possible final answers Just.Ask returns). As we can see, in 200 questions, Just.Ask attempted to answer 175 questions, 50.3% of the times correctly. Moreover, looking at the results of both Precision and Precision@1, we know that Just.Ask managed to push the correct answer to the top approximately 70.4% of the time.

# Questions	Correct	Wrong	No Answer
200	88	87	25

Accuracy	Precision	Recall	MRR	Precision@1
44%	50.3%	87.5%	0.37	35.4%

Table 1: Just.Ask baseline best results for the corpus test of 200 questions

In Tables 2 and 3, we show the results of the Answer Extraction module, using Levenshtein distance with a threshold of 0.33 to calculate the distance between candidate answers. For the different amounts of retrieved passages (8, 32 ou 64 passages) from Google, we present how successful were the Candidate Answer Extraction (CAE) and Answer Selection (AS) stages. The success of CAE stage is measured by the number of questions that manage to extract at least one correct answer, while the success of the AS stage is given by the number of questions that manage to extract one final answer from the list of candidate answers. They also present us a distribution of the number of questions with a certain amount of extracted answers.

	Google – 8 passages			Google – 32 passages		
# Extracted Answers	Questions (#)	CAE success	AS success	Questions (#)	CAE success	AS success
0	21	0	–	19	0	–
1 to 10	73	62	53	16	11	11
11 to 20	32	29	22	13	9	7
21 to 40	11	10	8	61	57	39
41 to 60	1	1	1	30	25	17
61 to 100	0	–	–	18	14	11
> 100	0	–	–	5	4	3
All	138	102	84	162	120	88

Table 2: Answer extraction intrinsic evaluation, using Google and two different amounts of retrieved passages.

We can see that this module achieves the best results with 64 retrieved passages. If we reduce the number of passages, the number of extracted candidates per question also diminishes. But if the number of passages is higher, we also risk losing precision, having way too many wrong answers in the mix. Moreover, the AS and CAE stages are 100% successful with 11 to 20 answers extracted. When that number increases, the accuracies do not show any concrete tendency to increase or decrease.

In Table 4, we show the evaluation results of the answer extraction component of Just.Ask according to the question category. The column 'Total' gives us the total number of questions per category, with 'Q.' being the number of questions for which at least one candidate answer was extracted. The success of the CAE and AS stages is defined in the same way as in the previous tables.

With the category Entity, the system only extracts one final answer, no

# Extracted Answers	Google – 64 passages Questions (#)	CAE success	AS success
0	17	0	0
1 to 10	18	9	9
11 to 20	2	2	2
21 to 40	16	12	8
41 to 60	42	38	25
61 to 100	38	32	18
> 100	35	30	17
All	168	123	79

Table 3: Answer extraction intrinsic evaluation, using 64 passages from the search engine Google.

		Google – 8 passages			Google – 32 passages			Google – 64 passages		
	Total	Q.	CAE	AS	Q.	CAE	AS	Q.	CAE	AS
ABB	4	3	1	1	3	1	1	4	1	1
DES	9	6	4	4	6	4	4	6	4	4
ENT	21	15	1	1	16	1	0	17	1	0
HUM	64	46	42	34	55	48	36	56	48	31
LOC	39	27	24	19	34	27	22	35	29	21
NUM	63	41	30	25	48	35	25	50	40	22
	200	138	102	84	162	120	88	168	123	79

Table 4: Answer extraction intrinsic evaluation, for different amounts of retrieved passages, according to the question category

matter how many passages are used. We found out that this category achieves the best result with only 8 passages retrieved from Google. Using 64 passages increases the number of questions, but at the same time, gives worse results for the AS section in three categories (Human, Location and Numeric) compared with the results obtained with 32 passages. This table also evidences the discrepancies between the different stages for all of these categories, showing that more effort should be put at a certain stage for a certain category – such as giving more focus to the Answer Selection, with the Numeric category, for instance.

3.3.2 Results with TREC Corpus

For the set of 1134 questions, the system answered correctly 234 questions (for a 20.6% accuracy) using Levenshtein metric with a threshold of 0.17 and with 20 paragraphs retrieved from New York Times, which represents an increase of approximately 32.2% (and more 57 correct answers than we had) over the previous approach of using the whole documents as passages. This paragraph indexing method proved to be much better overall than document indexing, and was therefore used permantly while searching for even better results.

These 1210 questions proved at least to have a higher accuracy and recall scores than the previous sets of questions, as we can see from the table 5 below.

	1134 questions	232 questions	1210 questions
Precision	26.7%	28.2%	27.4%
Recall	77.2%	67.2%	76.9%
Accuracy	20.6%	19.0%	21.1%

Table 5: Baseline results for the three sets of corpus that were created using TREC questions and New York Times corpus

Later, we tried to make further experiences with this corpus by increasing the number of passages to retrieve. We found out that this change actually had a considerable impact over the system extrinsic results, as Table 6 shows.

	20	30	50	100
Accuracy	21.1%	22.4%	23.7%	23.8%

Table 6: Accuracy results after the implementation of the features described in Sections 4 and 5 for 20, 30, 50 and 100 passages extracted from the NYT corpus

Even though the 100 passages offer the best results, it also takes more than thrice the time than running just 20, with a 2.7% difference in accuracy. So, depending on whether we prefer time over accuracy, any of these is viable, with the 30 passagens offering the best middle ground here.

3.4 Error Analysis over the Answer Module

This section was designed to report some of the most common mistakes that Just.Ask still presents over its Answer Module – mainly while clustering the answer candidates, failing to cluster equal or inclusive answers into the same cluster, but also at the answer extraction phase. These mistakes are directly linked to the distance algorithms we have at the moment (Levenshtein and Overlap metrics).

1. Wrong candidates extracted – for the question 'How long is the Okavango River', for instance, there were two answers of the Numeric type extracted: '1000 miles' and '300 km', with the former being the correct answer. Unfortunately, since there are only two instances of these answers, they both share the same score of 1.0. Further analysis of the passages containing the respective answers reveals that correctness:

   – 'The Okavango River rises in the Angolan highlands and flows over 1000 miles...'

– ‘Namibia has built a water canal, measuring about 300 km long, ... Siyabona Africa Travel (Pty) Ltd, ”Popa Falls — Okavango River — Botswana” webpage: ...’

As we can see, the second passage also includes the headword ‘Okavango River’, so we can not simply boost the score of the passage that includes the question’s headword...

Possible solution: but we can try to give more weight to a candidate answer if that answer is closer to the keywords used in the search by a certain predefined threshold.

2. Parts of person names/proper nouns – several candidates over categories such as Person and Location remain in different clusters, when they clearly refer the same entity. Here are a few examples:

   ‘Queen Elizabeth’ vs. ‘Elizabeth II’;

   ‘Mt. Kilimanjaro’ vs. ‘Mount Kilimanjaro’;

   ‘River Glomma’ vs ’Glomma’;

   ‘Consejo Regulador de Utiel-Requena’ vs. ‘Util-Requena’ vs. ‘Requena’;

   ‘Calvin Broadus’ vs. ‘Broadus’;

   Solution: we might extend a few set of hand-written rules to relate answers such as ‘Queen Elizabeth’ and ‘Elizabeth II’ or ‘Bill Clinton’ and ‘Mr. Clinton’, normalizing abbreviations such as ‘Mr.’, ‘Mrs.’, ‘Queen’, etc. Then, by applying a new metric that performs similarity on a token level, we might be able to cluster these answers. If there is at least one match between two tokens from candidate answers, then it is quite likely that the two mention the same entity, specially if both candidates share the same number of tokens and the innitials of the remaining tokens match. To give a simple example, consider the answers ‘J. F. Kennedy’ and ‘John Fitzgerald Kennedy’. A simple Levenshtein distance will detect a considerable number of edit operations. Should there be a better metric for these particular cases? LogSim distance takes into account the number of links, i.e. common tokens between two answers, which should give it a slight edge to Levenshtein. It would, if we had more than only one common link as we have here (‘Kennedy’) out of three words for each answer. So, we have LogSim approximately equal to 0.014. Levenshtein actually gives us a better score (0.59)!
   With the new proposed metric, we detect the common token ‘Kennedy’ and then we check the other tokens. In this case, we have an easy match, with both answers sharing the same number of words, and each word sharing at least the same initial, in the same order. What should be the threshold here, in case we do not have this perfect alignement? If, in another example, we have a single match between answers but the innitials of the other tokens are different, should we reject it? The answer is likely to be a ‘Yes.’.

3. Places/dates/numbers that differ in specificity – some candidates over the categories LOCATION and NUMERIC that might refer to a correct answer remain in separate clusters. In many cases, these are similar mistakes as the ones described in 2. (i.e. one location composes part of the name of the other), but instead of refering the same entity, these answers share a relation of inclusion. Examples:

   'Mt Kenya' vs. 'Mount Kenya National Park';

   'Bavaria' vs. 'Overbayern Upper Bavaria';

   '2009' vs. 'April 3, 2009'

   Moreover, for the question 'Where was the director Michelangelo Antonioni born', the answers 'Ferrara' and 'Italy' are found with equal scores (3.0). In this particular case, we can consider either of these answers as correct, with 'Ferrara' included in Italy (being a more specific answer). But a better answer to be extracted would probably be 'Ferrara, Italy'. The questions here are mainly: "Where do we draw a limit for inclusive answers? Which answer should we consider as representative if we group these answers in a single cluster?"

   Solution: based on [26], we decided to take into account relations of inclusion, and we are using the same rules that were used over the TREC corpus now with Just.Ask, hoping for similar improvements.

4. Language issue – many entities share completely different names for the same entity: we can find the name in the original language of the entity's location or in a normalized English form. And even small names such as 'Glomma' and 'Glåma' remain in separate clusters!

   Solution: Gazeteer information can probably help here. Specific metrics such as Soundex can also make a positive impact, even if in very particular cases, such as the 'Glomma' vs. 'Glåma example.

5. Different measures/Conversion issues – such as having answers in miles vs. km. vs. feet, pounds vs. kilograms, etc.

   Solution: create a normalizer that can act as a converter for the most important measures (distance, weight, etc.).

6. Misspellings/Formatting issues – many names/entities appear either misspelled or in different formats (capital letters vs. lower cases) over the web. While Levenshtein distance may handle most of these cases for the minimum threshold tested, there are still a few unresolved such as :

   'Kotashaan' vs. 'Khotashan';

   'Grozny' and 'GROZNY'

   Solution: the solution should simply be normalizing all the answers into a standard type. As for the misspelling issue, a similarity measure such as Soundex can solve more specific cases such as the one described above. The question remains: when and where do we apply it exactly?

7. Acronyms – seen in a flagrant example with the answers ‘U.S.’ and ‘US’ unclustered, but most generally pairs such as ‘United States of America’ and ‘U.S.A.’.

   Solution: In this case, we may extend abbreviations normalization to include this abbreviation.

Over this section, we have looked at several points of improvement. Next, we will describe how we solved all of these points.

4 Relating Answers

Over this section, we will try to solve the potential problems detected during the Error Analysis (see Section 3.4), using for that effort the implementation of the set of rules that will compose our 'relations module', new normalizers, and a new proximity measure.

4.1 Improving the Answer Extraction module: Relations Module Implementation

Figure 2 represents what we want for our system: to detect the relations described in the Section 2.1.

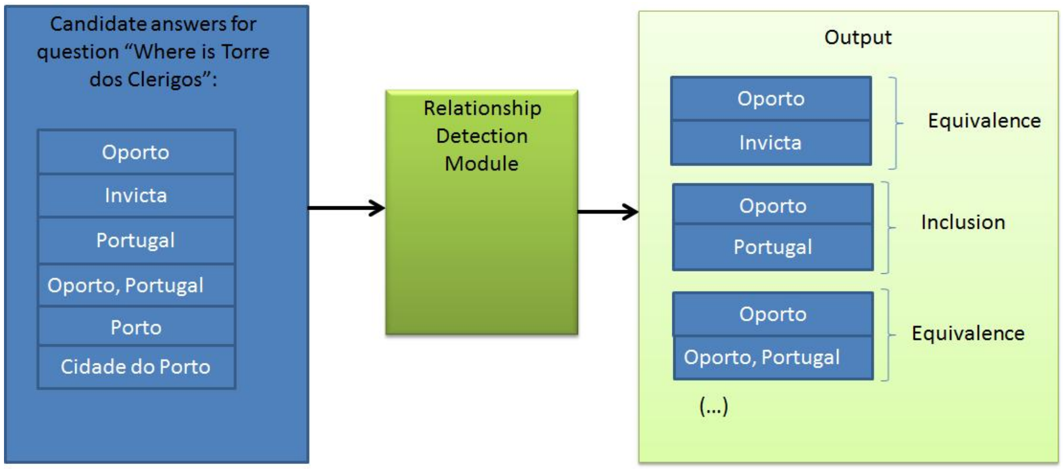


Figure 4: Proposed approach to automate the detection of relations between answers

We receive several candidate answers of a question as input and then, through what we can call a 'Relationship Detection Module', relationships between pairs of answers are assigned. This module will contain the metrics already available (Levenshtein and word overlap) plus the new metrics taken from the ideas collected over the related work. We also may want to assign different weights to different relationships, to value relationships of equivalence over inclusion, alternative and specially contradictory answers, which will possibly allow a better clustering of answers.

The goal here is to extend the metrics already existent to more efficient, semantically-based ones, using notions not only of synonymy, but also hiponymy and antonymy that we can access through WordNet. Moreover, we may want to analyze how much can the context that surronds an extracted answer influence the overall result, taking into account what we saw in [2]. We also plan to use the set of rules provided in [35] to enhance the final answer returned to the user.

After we have this process automatic and we have extended the answering module, we can perform a study on how relationships between answers can affect the outcome of the answer, and therefore the efficiency of the Answer Extraction module.

As it was described in Section 2.1, we define four types of relations between answers: Equivalence, Inclusion, Alternative, and Contradiction. For the context of aligning candidate answers that should share a same cluster, our first thought was that the equivalence relation had a much bigger weight than all the others. But inclusions should also matter, according to [26]. Moreover, they might even be more benefic than those of equivalence for a certain category.

The work in [26] has taught us that using relations between answers can improve considerably our results. In this step, we incorporated the rules used in the context of that paper to the Just.Ask system, and created what we proposed to do during the first part of this thesis – a module in Just.Ask to detect relations between answers and from then, use that for a (potentially) better clustering of answers, by boosting the score of candidates sharing certain relations. This would also solve point 3. described over section 3.4 ('Places/dates/numbers that differ in specificity').

These rules cover the following categories: Entity, Numeric, Location and Human: Individual. They can be seen as a set of clauses determining which relation we should assign to a pair of candidate answers.

The answers now possess a set of relations they share with other answers – namely the type of relation and the answer they are related to, and a score for that relation. The module itself is composed of an interface (Scorer) which supports all the rules for the three types of relations we are interested in counting (and associating to the respective answers) for a better clustering:

– Frequency – i.e., the number of times the exact same answer appears, without any variation;

– Equivalence;

– Inclusion (divided in two types, one for the answer that includes and the other for the answer that is included by the other)

For a first experiment, we decided to give a score of 0.5 to each relation of inclusion found for each answer and 1.0 to frequency and equivalence. Later, we decided to test the assessment made in [26] that relations of equivalence are actually more 'penalizing' than those of inclusion for the category Entity, and give a score of 0.5 to relations of equivalence and 1.0 to inclusions for the rules within that category. Results were not different at all for all the corpora.

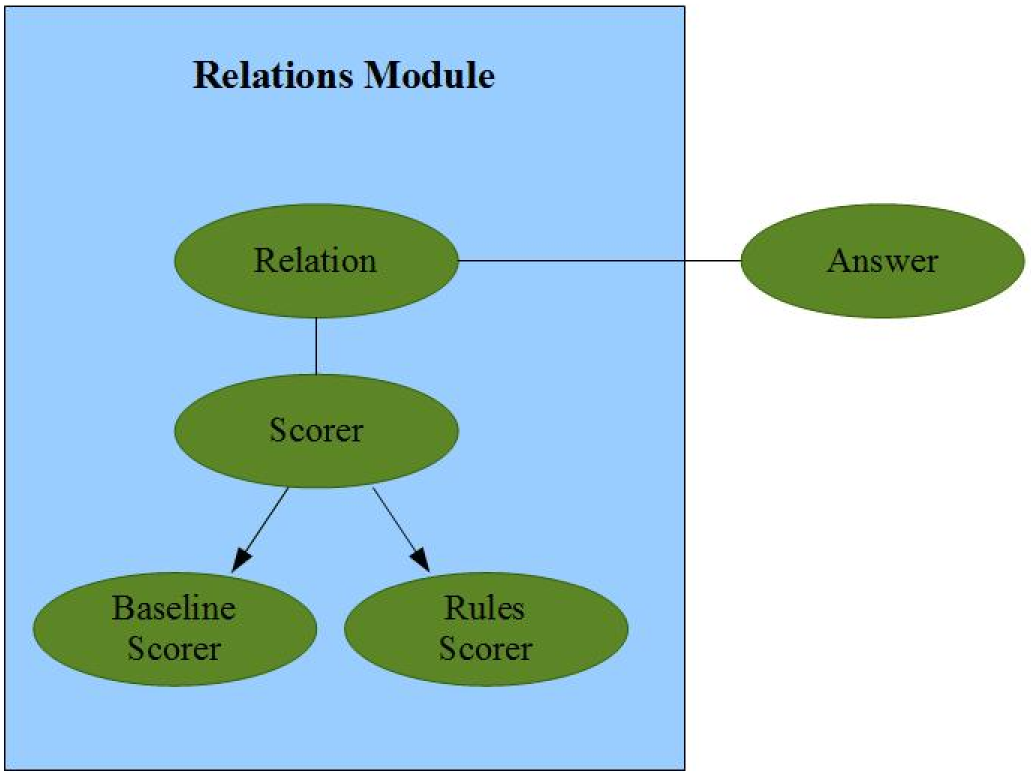


Figure 5: Relations' module basic architecture. Each answer has now a field representing the set of relations it shares with other answers. The interface Scorer, made to boost the score of the answers sharing a particular relation, has two components: Baseline Scorer, to count the frequency and Rules Scorer to count relations of equivalence and inclusion.

4.2 Additional Features

4.2.1 Normalizers

To solve the conversion, abbreviation and acronym issues described in points 2., 5. and 7. of Section 3.4., a few normalizers were developed. For the conversion aspect, a specific normalizer class was created dealing with answers of the type NUMERIC:DISTANCE, NUMERIC:SIZE, NUMERIC:WEIGHT and NUMERIC:TEMPERATURE that is able to convert kilometres to miles, meters to feet, kilograms to pounds and Celsius degrees to Fahrenheit, respectively.

Another normalizer was created to remove several common abbreviatures from the candidate answers, hoping that (with the help of new measures) we could improve clustering too. The abbreviatures removed were: 'Mr.', 'Mrs.', ' Ms.', 'King', 'Queen', 'Dame', 'Sir' and 'Dr.'. Over here, we find a particular issue worthy of noting: we could be removing something we should not be removing. Consider the possible answers 'Mr. T' or 'Queen' (the band). Curiously, an extended corpus of 1134 questions provided us even other example we would probably be missing: 'Mr. King' – in this case, however, we can easily limit King removals only to those occurences of the word 'King' right at the beggining of the answer. We can also add exceptions for the cases above. Still, it remains a curious statement that we must not underestimate, under any circumstances, the richness of a natural language.

A normalizer was also made to solve the formatting issue of point 6., to treat 'Grozny' and 'GROZNY' as equal answers.

For the point 7., a generic acronym normalizer was used, converting acronyms such as 'U.S.' to 'United States', or 'UN' to 'United Nations' – for categories such as LOCATION:COUNTRY and HUMAN:GROUP. As with the case of the normalizers above, it did not produce any noticeable changes in the system results, but it also evaluates very particular cases for the moment, allowing room for further extensions.

4.2.2 Proximity Measure

As with the problem in 1., we decided to implement the solution of giving extra weight to answers that remain closer to the query keywords that appear through the passages. For that, we calculate the average distance from the answer to the several keywords, and if that average distance is under a certain threshold (we defined it as being 20 characters long), we boost that answer's score (by 5 points).

4.3 Results

	200 questions	1210 questions*
Before features	P:50.6% R:87.0% A:44.0%	P:28.0% R:79.5% A:22.2%
After features	A:51.1% R:87.0% A:44.5%	P:28.2% R:79.8% A:22.5%

Table 7: Precision (P), Recall (R)and Accuracy (A) Results before and after the implementation of the features described in 4.1, 4.2 and 4.3. *Using 30 passages from New York Times for the corpus of 1210 questions from TREC.

4.4 Results Analysis

These preliminary results remain underwhelming, taking into account the effort and hope that was put into the relations module (Section 4.1) in particular.

The variation in results is minimal – only one to three correct answers, depending on the corpus test (Multisix or TREC).

4.5 Building an Evaluation Corpus

To evaluate the 'relation module' described over 4.1, a corpus for evaluating relations between answers was designed, before dealing with the implementation of the module itself. For that effort, we took the references we gathered in the previous corpus, and for each pair of references, assign one of the possible four types of relation they share (Equivalence, Inclusion, Contradiction or Alternative). A few of these relations were not entirely clear, and in that case, a fifth option ('In Doubt') was created. This corpus was eventually not used, but it remains here for future work purposes.

As it happened with the 'answers corpus', in order to create this corpora containing the relationships between the answers more efficiently, a new application was made. It receives the questions and references file as a single input, and then goes to iterate on the references on pairs, asking the user to choose one of the following options, corresponding to the relations mentioned in Section 2.1:

– 'E', for a relation of equivalence between pairs;

– 'I', if it is a relation of inclusion (this was later expanded to 'I1' if the first answer includes the second, and 'I2' if the second answer includes the first);

– 'A' for alternative (complementary) answers;

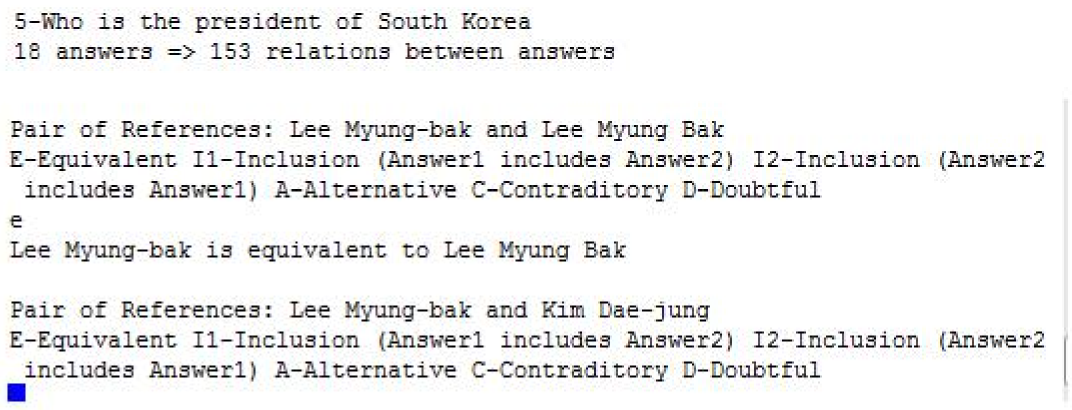
– 'C' for contradictory answers;

– ‘D’ for causes of doubt between relation types.

If the command given by the input fails to match any of these options, the script asks the user to repeat the judgment. If the question only has a single reference, it just advances to the next question. If the user running the script wants to skip a pair of references, he/she can type ‘skip’. At the end, all the relations are printed to an output file too. Each line has: 1) the identifier of the question; 2) the type of relationship between the two answers (INCLUSION, EQUIVALENCE, COMPLEMENTARY ANSWERS, CONTRADITORY ANSWERS or IN DOUBT); and 3) the pair of answers evaluated. For example, we have:

QUESTION 5 : EQUIVALENCE -Lee Myung-bak AND Lee

In other words, for the 5th question (‘Who is the president of South Korea’), there is a relationship of equivalence between the answers/references ‘Lee Myung-bak’ and ‘Lee’.



```
5-Who is the president of South Korea
18 answers => 153 relations between answers


Pair of References: Lee Myung-bak and Lee Myung Bak
E-Equivalent I1-Inclusion (Answer1 includes Answer2) I2-Inclusion (Answer2
 includes Answer1) A-Alternative C-Contraditory D-Doubtful
e
Lee Myung-bak is equivalent to Lee Myung Bak

Pair of References: Lee Myung-bak and Kim Dae-jung
E-Equivalent I1-Inclusion (Answer1 includes Answer2) I2-Inclusion (Answer2
 includes Answer1) A-Alternative C-Contraditory D-Doubtful
```

Figure 6: Execution of the application to create relations’ corpora in the command line

Over the next chapter, we will try to improve the results working directly with several possible distance measures (and combinations of two or more of these) we can apply during the clustering phase.

5 Clustering Answers

As we concluded in Section 4.4., results were still not improving as much as one can hope for. But we still had new clustering metrics to potentially improve our system, as we proposed also doing. Therefore, we decided to try new experiments, by bringing new similarity metrics to the clustering stage. The main motivation was to find a new measure (or a combination of measures) that could bring better results than Levenshtein did for the baseline.

5.1 Testing Different Similarity Metrics

5.1.1 Clusterer Metrics

Our practical work here began with experimenting different similarity metrics, integrating them into the distances package, which contained the Overlap and Levenshtein distances. For that, several lexical metrics that were used for a Natural Language project in 2010/2011 at Tagus Park were added. These metrics were taken from a Java similarity package (library) – 'uk.ac.shef.wit.simmetrics.similaritymetrics':

- Jaccard Index [11] – the Jaccard index (also known as Jaccard similarity coefficient) is defined by the size of the intersection divided by the size of the union between two sets (A and B):

$$J(A, B) = \frac{|A \cap B|}{|A \cup B|} \tag{15}$$

  In other words, the idea will be to divide the number of characters in common by the sum of the total number of characters of the two sequences;

- Dice's Coefficient [6] – related to Jaccard, but the similarity is now computed using the double of the number of characters that the two sequences share in common. With X and Y as two sets of keywords, Dice's coefficient is given by the following formula:

$$Dice(A, B) = \frac{2|A \cap B|}{|A| + |B|} \tag{16}$$

  Dice's Coefficient will basically give the double of weight to the intersecting elements, doubling the final score. Consider the following two strings: 'Kennedy' and 'Keneddy'. When taken as a string similarity measure, the coefficient can be calculated using the sets of bigrams for each of the two strings [15]. The set of bigrams for each word is {Ke, en, nn, ne, ed, dy} and {Ke, en, ne, ed, dd, dy}, respectively. Each set has six elements, and the intersection of these two sets gives us exactly four elements: {Ke, en, ne, dy}. The result would be (2*4)/(6+6) = 8/12 = 2/3. If we were using Jaccard, we would have 4/12 = 1/3!

– Jaro [13, 12] – this distance is mainly used in the record linkage (duplicate detection) area. It is calculated by the following:

$$Jaro(s1, s2) = \frac{1}{3}(\frac{m}{|s1|} + \frac{m}{|s2|} + \frac{m-t}{m}) \tag{17}$$

in which s1 and s2 are the two sequences to be tested, $|s1|$ and $|s2|$ are their lengths, m is the number of matching characters in s1 and s2 and t is the number of transpositions needed to put the matching characters in the same order.

– Jaro-Winkler [45] – is an extension of the Jaro metric. This distance is given by:

$$dw = dj + (lp(1 - dj)) \tag{18}$$

with dj being the Jaro distance between the two sequences, l the length of the common prefix and p a constant scaling factor (the standard value for this constant in Winkler's work is p = 0.1). Jaro-Winkler therefore puts more weight into sharing a common prefix, compared to the original Jaro distance. In other words, it will give a much better score to strings such as 'Arnold' vs. 'Arnie' or misspellings such as 'Martha' vs. 'Marhta' than Jaro.

– Needleman-Wunsch [30] – dynamic programming algorithm, similar to Levenshtein distance, but adding a variable cost G (instead of the constant 1 for Levenshtein):

$$D(i, j) = \begin{cases} D(i-1, j-1) + G \\ D(i-1, j) + G \\ D(i, j-1) + G \end{cases} \tag{19}$$

This package contained other metrics of particular interest to test, such as the following two measures, which were also added and tested:

– Soundex [36] – With this algorithm more focused to match words which sound alike, but differ lexically, we can expect to match misspelled answers with a higher Levenshtein distance than the threshold proposed (such as 'Massachussets' vs. 'Machassucets'), or even match answers in different languages that happen to sound alike, but spelled differently.

– Smith-Waterman [40] – is a variation of Needleman-Wunsch, commonly used in bioinformatics to align protein or nucleotide sequences. It compares segments of all possible lengths (local alignments) instead of looking at the sequence in its entirety, and chooses the one that maximises the similarity measure, leading us to believe that it might work better with answers with common affixes, such as 'acting' and 'enact'.

Then, the LogSim metric by Cordeiro and his team, described over the Related Work in Section 3.2.2, was implemented and added to the package, using the same weighting penalty of 2.7 as in the team's experiments. We then tested all the similarity measures using three different thresholds values: 1/2 (0.5), 1/3(0.33) and 1/6(0.17) of the single-link clustering algorithm, in order to evaluate how much does it affect the performance of each metric.

5.1.2 Results

Results are described in Table 8. We decided to evaluate these metrics using the corpus of 200 questions for a simple reason: it is by far the corpus that provides us the best results of the system at this point, and we should expect proporcional results between these two corpus either way.

5.1.3 Results Analysis

As we can see, the metric that achieves the best results is still Levenshtein distance, with the LogSim distance at a close second place overall – even holding the distintion of being the best measure for the highest threshold evaluated (0.5). Defining the 'best' threshold was not as clear as picking the best metric given these results. The threshold of 0.17 provides the best results for more than half of the metrics here, but there are cases when it is only the second best, even if the difference itself is virtually inexistent in most cases.

Metric/Threshold	**0.17**	**0.33**	**0.5**
Levenshtein	**P:51.1%** **R:87.0%** **A:44.5%**	P:43.7% R:87.5% A:38.5%	P:37.3% R:84.5% A:31.5%
Overlap	P:37.9% R:87.0% A:33.0%	P:33.0% R:87.0% A:33.0%	P:36.4% R:86.5% A:31.5%
Jaccard index	P:9.8% R:56.0% A:5.5%	P:9.9% R:55.5% A:5.5%	P:9.9% R:55.5% A:5.5%
Dice Coefficient	P:9.8% R:56.0% A:5.5%	P:9.9% R:56.0% A:5.5%	P:9.9% R:55.5% A:5.5%
Jaro	P:11.0% R:63.5% A:7.0%	P:11.9% R:63.0% A:7.5%	P:9.8% R:56.0% A:5.5%
Jaro-Winkler	P:11.0% R:63.5% A:7.0%	P:11.9% R:63.0% A:7.5%	P:9.8% R:56.0% A:5.5%
Needleman-Wunch	P:43.7% R:87.0% A:38.0%	P:43.7% R:87.0% A:38.0%	P:16.9% R:59.0% A:10.0%
LogSim	P:43.7% R:87.0% A:38.0%	P:43.7% R:87.0% A:38.0%	A:42.0% R:87.0% A:36.5%
Soundex	P:35.9% R:83.5% A:30.0%	P:35.9% R:83.5% A:30.0%	P:28.7% R:82.0% A:23.5%
Smith-Waterman	P:17.3% R:63.5% A:11.0%	P:13.4% R:59.5% A:8.0%	P:10.4% R:57.5% A:6.0%

Table 8: Precision (P), Recall (R) and Accuracy (A) results for every similarity metric added, for a given threshold value, using the test corpus of 200 questions and 32 passages from Google. Best precision and accuracy of the system bolded.

5.2 New Distance Metric - 'JFKDistance'

5.2.1 Description

Considering the two candidate answers 'J.F. Kennedy' and 'John Fitzgerald Kennedy', Levenshtein will give us a very high distance, and therefore will not group these two answers into a single cluster. The goal here is to finally be able to relate answers such as these. A new measure was therefore created in order to improve the overall accuracy of Just.Ask. This measure is in a way, a progression from metrics such as Overlap Distance and LogSim, also using a tokenizer to break the strings into a set of tokens, and then taking the basic concept of common terms (we can also call it links, words, or tokens) between the two strings and also adding an extra weight to strings that not only share at least a perfect match but have also another token that shares the first letter with the other token from the other answer. To exemplify, consider the example above. We have a perfect match in the common term Kennedy, and then we have two tokens unmatched. Even the apparently intuitive Overlap distance will give us a value of 2/3, making it impossible to cluster these two answers. Levenshtein will be closer to 0.5. Still underwhelming. Since we already have one word in common, with the new metric, we can assume with some security that there is a relation between 'John' and 'J.' and 'Fitzgerald' and 'F.'. This weight will obviously not be as big as the score we give to a 'perfect' match, but it should help us clustering more of these types of answers. For that matter, we divide by 2 what would be a value identical to a match between two tokens for that extra weight.

The new distance is computed the following way:

$$distance = 1 - commonTermsScore - firstLetterMatchesScore \tag{20}$$

in which the scores for the common terms and matches of the first letter of each token are given by:

$$commonTermsScore = \frac{\#commonTerms}{max(length(a1), length(a2))}; \tag{21}$$

and

$$firstLetterMatchesScore = \frac{\frac{\#firstLetterMatches}{nonCommonTerms}}{2} \tag{22}$$

with a1 and a2 being the two answers converted to strings after normalization, and nonCommonTerms being the terms that remain unmatched, such as 'J.', 'John', 'F.' and'Fitzgerald' in the above example.

A few flaws about this measure were thought later on, namely the scope of this measure whithin our topology of answers: if, for example, we apply this same measure to answers of type NUMERIC, results could become quite unpredictable, and wrong. It became quite clear that a separation of measures given a type of answer (NUMERIC, ENTITY, LOCATION, etc.) was needed, if we wanted to use this measure to its full potential, as we will see.

5.2.2 Results

Table 9 shows us the results after the implementation of this new distance metric.

	200 questions	1210 questions*
Before	P:51.1% R:87.0% A:44.5%	P:28.2% R:79.8% A:22.5%
After new metric	P:51.7% R:87.0% A:45.0%	P:29.2% R:79.8% A:23.3%

Table 9: Precision (P), Recall (R) and Accuracy (A) Results after the implementation of the new distance metric. *Using 30 passages from New York Times for the corpus of 1210 questions from TREC.

5.2.3 Results Analysis

As we can see for these results, even with the redundancy of sharing features that we also have in the rules and in certain normalizers, this metric can actually improve our results even if the difference is minimal, and not statistically significant.

5.3 Possible new features – Combination of distance metrics to enhance overall results

We also thought in analyzing these results by the question category, using an evaluation framework developed earlier. The motivation behind this evaluation was to build a decision tree of sorts, that would basically choose a metric (or a conjunction of metrics) for a certain category (or a group of categories). If, for instance, we detect that the Levenshtein distance gives us the best answers for the category HUMAN but for the category LOCATION, LogSim was the metric which offered us the best results, we would have something like:

if ($Category == Human$) use Levenshtein

if ($Category == Location$) use LogSim

(...)

For that effort, we used the best thresholds for each metric, using the corpus of 200 questions and Google search engine, retrieving 32 passages. Our question classifier achieves an accuracy of 90% [39], and there have been a few answers not labeled as correct by the framework that are indeed correct. With that in mind, and considering that these are issues that are common to every single distance metric used (so they should not affect that much the analysis we are looking for), here are the results for every metric incorporated by the three main categories Human, Location and Numeric that can be easily evaluated (categories such as Description, by retrieving the Wikipedia abstract, are just marked as wrong/nil, given the reference input), shown in Table 10:

Metric/Category	**Human**	**Location**	**Numeric**
Levenshtein	50.00%	**53.85%**	**4.84%**
Overlap	42.42%	41.03%	**4.84%**
Jaccard index	7.58%	0.00%	0.00%
Dice	7.58%	0.00%	0.00%
Jaro	9.09%	0.00%	0.00%
Jaro-Winkler	9.09%	0.00%	0.00%
Needleman-Wunch	42.42%	46.15%	**4.84%**
LogSim	42.42%	46.15%	**4.84%**
Soundex	42.42%	46.15%	3.23%
Smith-Waterman	9.09%	2.56%	0.00%
JFKDistance	**54.55%**	**53.85%**	**4.84%**

Table 10: Accuracy results for every similarity metric added, for each category, using the test corpus of 200 questions and 32 passages from Google. Best precision and accuracy of the system bolded, except for when all the metrics give the same result.

We can easily notice that all of these metrics behave in similar ways through each category, i.e., results are always 'proporcional' in each category to how they perform overall. That stops us from picking best metrics for each category, since we find out that the new distance implemented equals or surpasses (as with the case of Human category) all the others.

5.4 Posterior analysis

In another effort to improve the results above, a deeper analysis was made through the system's output regarding the corpus test of 200 questions, in order to detect certain 'error patterns' that could be resolved in the future. Below are the main points of improvement:

- Time-changing answers – Some questions do not have a straight answer, one that never changes with time. A clear example is a question such as 'Who is the prime-minister of Japan'. We have observed that former prime-minister Yukio Hatoyama actually earns a higher score than current prime-minister Naoto Kan, which should be the correct answer. The solution: Give an extra weight to passages that are at least from the same year of the collected corpus from Google? (in this case: 2010).

- Numeric answers are easily failable due to the fact that when we expect a number, it can pick up words such as 'one', actually retrieving that as a top answer in a couple of cases. Moreover, when it expects an answer of type NUMERIC:DATE, it could still easely choose the date/year when the article was posted (or even the number of views of that article!). The solution might pass for: a) ignoring 'one' to 'three', when written in words; b) exclude the publishing date from the snippet somehow; and c) improve the regular expressions used to filter candidates.

– Filtering stage does not seem to work in certain questions – in some questions, we found that one or more of the final answers are actually within the question, under certain variations. For instance, for the question 'Who is John J. Famalaro accused of having killed', John Famalaro is the top answer retrieved... The solution here should be applying a set of rules to remove these variations from the final answers.

– Answers in different formats than expected – some questions imply multiple answers, but we are only searching for a single one. Here is a good example: the correct answer for the question 'When did Innsbruck host the Winter Olympics' is actually two dates, as it is stated in the reference file, but the system only goes to retrieve one of them, leading to a incorrect answer. Should we put the multiple answers separated as single references? Other questions, such as 'When did the journalist Charlotte Bass live', demand a time interval and yet a single year is retrieved. What to do in this case?

– Difficulty tracing certain types of answer (category confusion?) – for the question 'What position does Lothar Matthaus play' the answer 'Germany' appears on top! In another example of not finding the right type of answer, the question 'What was the first film directed by John Milius' is supposed to answer 'Dillinger' but chooses 'western' and 'documentary' as answers instead. We know that the question classifier is not 100% foul proof, but maybe we should try to make a list of acceptable terms for certain headwords? And in the case of films and other works, perhaps trace only those between "", if possible.

– Incomplete references – the most easily detectable error to be corrected is the reference file, that presents us all the possible 'correct' answers, potentially missing viable references.

After the insertion of gaps such as this, while also removing a contradictory reference or two, the results improved greatly, and the system now manages to answer correctly 96 of the 200 questions for an accuracy of 48.0% and a precision of 55.2%.

6 Conclusions and Future Work

Over the course of this thesis, we have experimented new ways of improving the Answers Module of Just.Ask, our own QA system.

Here are the main contributions achieved:

– We performed a state of art on relations between answers' relations and paraphrase identification, to detect new possible ways to improve Just.Ask's results with the Answer Extraction module.

– We implemented a new relations module that boosts the score of candidate answers sharing relations of equivalence and inclusion;

– We developed new metrics to improve the clustering of answers;

– We developed new answer normalizers to relate two equal answers in different formats;

– We developed new corpora that we believe will be extremely useful in the future;

– We performed further analysis to detect persistent error patterns.

In the end, results were not as good as we could hope for, but we still managed to improve the system's precision as high as 9.1%, using a modified version of the Multisix corpus composed of 200 questions.

There is still many work to be done in the future. We understand that the problem here does not lie exclusively with the Answer module of Just.Ask – what happens before has a severe impact over this module, namely during query formulation and passage retrieval. Over the many tasks we should try, here are a selected few:

– Develop a more semantic clusterer, instead of merely 'ortographical', by adding more rules that deal with the semantic of the answers;

– Use the rules as a distance metric of its own;

– Finally, implementing solutions to the problems and questions unanswered over Section 5.4 can help improving somewhat these results.

References

[1] S. Banerjee and T. Pedersen. Extended gloss overlaps as a measure of semantic relatedness. In *Proc. of the 18th Int'l. Joint Conf. on Artificial Intelligence*, pages 805–810, 2003.

[2] Regina Barzilay and Noemie Elhadad. Sentence alignment for monolingual comparable corpora. In *In Proceedings of the 2003 conference on Empirical methods in natural language processing*, pages 25–32, 2003.

[3] Regina Barzilay and Lillian Lee. Learning to paraphrase: An unsupervised approach using multiple-sequence alignment. In *Proceedings of the Joint Human Language Technology/North American Chapter of the ACL Conference (HLT-NAACL)*, pages 16–23, 2003.

[4] Marie catherine De Marneffe, Bill Maccartney, and Christopher D. Manning. Generating typed dependency parses from phrase structure parses. In *In LREC 2006*, 2006.

[5] Joao Cordeiro, Gael Dias, and Pavel Brazdil. Unsupervised learning of paraphrases. In *In Research in Computer Science. National Polytechnic Institute, Mexico. ISSN*, pages 1870–4069, 2007.

[6] L. Dice. Measures of the amount of ecologic association between species. *Ecology*, 26(3):297–302, 1945.

[7] Bill Dolan, Chris Quirk, and Chris Brockett. Unsupervised construction of large paraphrase corpora: Exploiting massively parallel news sources. In *In Proceedings of the 20th International Conference on Computational Linguistics*, 2004.

[8] S. Fernando and M. Stevenson. A semantic similarity approach to paraphrase detection. In *In CLUCK 2008*, pages 18–26, 2008.

[9] Vasileios Hatzivassiloglou, Judith L. Klavans, and Eleazar Eskin. Detecting text similarity over short passages: Exploring linguistic feature combinations via machine learning, 1999.

[10] D. Hays. Dependency theory: A formalism and some observations. *Languages, 40:*, pages 511–525, 1964.

[11] P. Jaccard. Distribution de la flore alpine dans le bassin des Dranses et dans quelques régions voisines. *Bulletin de la Société Vaudoise des Sciences Naturelles*, 37:241–272, 1901.

[12] M. A. Jaro. Probabilistic linkage of large public health data file. In *Statistics in Medicine*, volume 14, pages 491–498, 1995.

[13] Matthew A. Jaro. Advances in record-linkage methodology as applied to matching the 1985 census of Tampa, Florida. *Journal of the American Statistical Association*, 84(406):414–420, 1989.

[14] J.J. Jiang and D.W. Conrath. Semantic similarity based on corpus statistics and lexical taxonomy. In *Proc. of the Int'l. Conf. on Research in Computational Linguistics*, pages 19–33, 1997.

[15] Grzegorz Kondrak, Daniel Marcu, and Kevin Knight. Cognates can improve statistical translation models. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology: companion volume of the Proceedings of HLT-NAACL 2003–short papers - Volume 2*, NAACL-Short '03, pages 46–48. Association for Computational Linguistics, 2003.

[16] Zornitsa Kozareva, Sonia Vázquez, and Andrés Montoyo. The effect of semantic knowledge expansion to textual entailment recognition. In *TSD*, volume 4188 of *Lecture Notes in Computer Science*, pages 143–150. Springer, 2006.

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

[18] C. Leacock and M. Chodorow. Combining local context and wordnet similarity for word sense identification. In *MIT Press*, pages 265–283, 1998.

[19] Vladimir Iosifovich Levenshtein. Binary codes capable of correcting deletions, insertions and reversals. *Soviet Physics Doklady*, 10(8):707–710, 1966. Doklady Akademii Nauk SSSR, V163 No4 845-848, 1965.

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

[21] Dekang Lin. Principle-based parsing without overgeneration. In *ACL*, pages 112–120, 1993.

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

[23] Mihai C. Lintean and Vasile Rus. Paraphrase identification using weighted dependencies and word semantics. In H. Chad Lane and Hans W. Guesgen, editors, *FLAIRS Conference*. AAAI Press, 2009.

[24] Bernardo Magnini, Simone Romagnoli, Alessandro Vallin, Jesús Herrera, Anselmo Penas, Víctor Peinado, Felisa Verdejo, and Maarten de Rijke. The multiple language question answering track at clef 2003. In *CLEF*, volume 3237 of *Lecture Notes in Computer Science*, pages 471–486. Springer, 2003.

[25] Bernardo Magnini, Alessandro Vallin, Christelle Ayache, Gregor Erbach, Anselmo Pe nas, Maarten de Rijke, Paulo Rocha, Kiril Ivanov Simov, and Richard F. E. Sutcliffe. Overview of the clef 2004 multilingual question answering track. In *CLEF*, volume 3491 of *Lecture Notes in Computer Science*, pages 371–391. Springer, 2004.

[26] A. Mendes and L. Coheur. An approach to answer selection in question answering based on semantic relations, 2011. Accepted for publication in IJCAI 2011, the 22nd International Joint Conference on Artificial Intelligence.

[27] Rada Mihalcea, Courtney Corley, and Carlo Strapparava. Corpus-based and knowledge-based measures of text semantic similarity. In *In AAAI'06*, pages 775–780, 2006.

[28] George A. Miller. Wordnet: A lexical database for english. *Communications of the ACM*, 38:39–41, 1995.

[29] Véronique Moriceau. Numerical data integration for cooperative question-answering, 2006.

[30] Saul B. Needleman and Christian D. Wunsch. A general method applicable to the search for similarities in the amino acid sequence of two proteins. *Journal of Molecular Biology*, 48(3):443–453, 1970.

[31] Franz Josef Och and Hermann Ney. A systematic comparison of various statistical alignment models. *Computational Linguistics*, 29, 2003.

[32] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Computational Linguistics (ACL), Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL), Philadelphia*, pages 311–318, 2002.

[33] Long Qiu, Min-Yen Kan, and Tat-Seng Chua. Paraphrase recognition via dissimilarity significance classification. In *EMNLP*, pages 18–26. ACL, 2006.

[34] Philip Resnik. Using information content to evaluate semantic similarity in a taxonomy. In *Proc. of the 14th Int'l. Joint Conf. on Artificial Intelligence (AAAI)*, pages 448–453, 1995.

[35] Joana Ribeiro. Final report in question-answer systems, 2010. Universidade Técnica de Lisboa – Instituto Superior Técnico, Portugal.

[36] R. C. Russell. Soundex phonetic comparison system [cf.u.s. patents 1261167 (1918), 1435663 (1922)], 1918.

[37] Gerard Salton and Christopher Buckley. Term-weighting approaches in automatic text retrieval. In *INFORMATION PROCESSING AND MANAGEMENT*, pages 513–523, 1988.

[38] J. Silva. QA+ML@Wikipedia&Google. Master's thesis, Universidade Técnica de Lisboa – Instituto Superior Técnico, Portugal, 2009.

[39] J. Silva, L. Coheur, and A. Mendes. Just ask–A multi-pronged approach to question answering, 2010. Submitted to Artificial Intelligence Tools.

[40] T. Smith and M. Waterman. Identification of common molecular subsequences. *Journal of Molecular Biology*, 147(1):195–197, 1981.

[41] Mark Stevenson and Mark A. Greenwood. A semantic approach to IE pattern induction. In *ACL*. The Association for Computer Linguistics, 2005.

[42] Peter D. Turney. Mining the web for synonyms: PMI-IR versus LSA on TOEFL. In *EMCL '01: Proceedings of the 12th European Conference on Machine Learning*, pages 491–502. Springer-Verlag, 2001.

[43] Ellen M. Voorhees. Overview of TREC 2003. In *TREC*, pages 1–13, 2003.

[44] Bonnie Webber, Claire Gardent, and Johan Bos. Position statement: Inference in question answering. In *Third international conference on Language Resources and Evaluation - LREC'02*, Las Palmas, Spain, 2002.

[45] William E. Winkler. String comparator metrics and enhanced decision rules in the fellegi-sunter model of record linkage. In *Proceedings of the Section on Survey Research*, pages 354–359, 1990.

[46] Zhibiao Wu and Martha Palmer. Verb semantics and lexical selection. In *Proc. of the 32nd annual meeting on Association for Computational Linguistics*, pages 133–138, 1994.

[47] Yitao Zhang and Jon Patrick. Paraphrase identification by text canonicalization. In *Proceedings of the Australasian Language Technology Workshop 2005*, pages 160–166, Sydney, Australia, 2005.

Appendix

Appendix A – Rules for detecting relations between rules