Coling 2008

22nd International Conference on Computational Linguistics

Proceedings of the 2nd workshop on Information Retrieval for Question Answering

Workshop chair:
Mark A. Greenwood

24 August 2008
Manchester, UK

Order copies of this and other Coling proceedings from:

Association for Computational Linguistics (ACL)
209 N. Eighth Street
Stroudsburg, PA 18360
USA
Tel: +1-570-476-8006
Fax: +1-570-476-0860
acl@aclweb.org

ISBN 978-1-905593-55-2

Design by Chimney Design, Brighton, UK
Production and manufacture by One Digital, Brighton, UK

Introduction

Open domain question answering (QA) has become a very active research area over the past decade, due in large measure to the stimulus of the TREC Question Answering track (now a track within the recently formed Text Analysis Conference, TAC). This track addresses the task of finding **answers** to natural language questions (e.g. "How tall is the Eiffel Tower?", "Who is Aaron Copland?", "What effect does second-hand smoke have on non-smokers?") from large text collections. This task stands in contrast to the more conventional information retrieval (IR) task of finding **documents** relevant to a query, where the query may be simply a collection of keywords (e.g. "Eiffel Tower", "American composer, born Brooklyn NY 1900, ...").

Finding answers requires processing texts at a level of detail that cannot be carried out at retrieval time for very large text collections. This limitation has led many researchers to rely on, broadly, a two stage approach to the QA task. In stage one a subset of question-relevant texts are selected from the whole collection. In stage two this subset is subjected to detailed processing for answer extraction. Clearly performance at stage two is bounded by performance at stage one, and previous work has shown that, despite the sophistication of standard IR ranking algorithms, they are not well suited to the stage one task of retrieving relevant documents given short natural language questions. It is likely that improvements in this area will come from linguistic insights into why QA focused IR is different from the traditional IR model.

With the continued expansion of QA research into more complex question types and with the speed with which answers are returned becoming an issue, the importance of having good, QA-focused IR techniques is likely to increase. To date this topic has received limited explicit attention despite its obvious importance. This 2nd IR4QA workshop aims to address this situation by continuing to attract the attention of researchers to the specific IR challenges raised by QA.

For this workshop, we solicited papers that addressed any aspect of QA-focused IR, in order to improve overall system performance, , suggesting possible topics such as:

- parameterizations/optimizations of specific IR systems for QA
- studies of query formation strategies suited to QA, e.g. named entity pre-processing of questions
- different uses of IR for different question types (e.g. factoid, list, definition, event, how, ...)
- utility of term matching constraints, e.g. term proximity, for QA
- analyses of differing IR techniques for QA
- impact of IR performance on overall QA performance
- QA-orientated corpus pre-processing, e.g. indexing POS tags, named entities, semantically-tagged entities, relationships, etc. rather than simply tokens
- evaluation measures for assessing IR for QA
- retrieval from semi-structured data - i.e. QA from Wikipedia articles

From the papers submitted, 10 were selected following peer review. These papers are included in this proceedings. The enthusiastic response to this workshop confirms the belief that this is an important area of interest to a significant number of researchers.

Mark A. Greenwood

Organizers:

Mark A. Greenwood, Univeristy of Sheffield

Programme Committee:

Matthew W. Bilotti, Carnegie Mellon University
Gosse Bouma, University of Groningen
Charles Clarke, University of Waterloo
Hoa Dang, NIST
Robert Gaizauskas, University of Sheffield
Eduard Hovy, ISI
Jimmy Lin, University of Maryland
John Prager, IBM
Horacio Saggion, University of Sheffield
Jrg Tiedemann, University of Groningen
Bonnie Webber, University of Edinburgh
Ralph Weischedel, BBN

Table of Contents

Conference Programme

Sunday, August 24, 2008

9:15–9:30 Welcome

9:30–10:00 *Improving Text Retrieval Precision and Answer Accuracy in Question Answering Systems*
Matthew Bilotti and Eric Nyberg

10:00–10:30 *Exact Phrases in Information Retrieval for Question Answering*
Svetlana Stoyanchev, Young Chol Song and William Lahti

10:30–11:00 Coffee Break

11:00–11:30 *Simple is Best: Experiments with Different Document Segmentation Strategies for Passage Retrieval*
Jörg Tiedemann and Jori Mur

11:30–12:00 *Passage Retrieval for Question Answering using Sliding Windows*
Mahboob Khalid and Suzan Verberne

12:00–12:30 *A Data Driven Approach to Query Expansion in Question Answering*
Leon Derczynski, Jun Wang, Robert Gaizauskas and Mark A. Greenwood

12:30–2:00 Lunch

2:00–2:30 *Answer Validation by Information Distance Calculation*
Fangtao Li, Xian Zhang and Xiaoyan Zhu

2:30–3:00 *Using Lexico-Semantic Information for Query Expansion in Passage Retrieval for Question Answering*
Lonneke van der Plas and Jörg Tiedemann

3:00–3:30 *Evaluation of Automatically Reformulated Questions in Question Series*
Richard Shaw, Ben Solway, Robert Gaizauskas and Mark A. Greenwood

3:30–4:00 Coffee Break

4:00–4:30 *Topic Indexing and Retrieval for Factoid QA*
Kisuh Ahn and Bonnie Webber

4:30–5:00 *Indexing on Semantic Roles for Question Answering*
Luiz Augusto Pizzato and Diego Mollá

5:00–5:30 Discussion Session

Improving Text Retrieval Precision and Answer Accuracy in Question Answering Systems

Matthew W. Bilotti and Eric Nyberg
Language Technologies Institute
Carnegie Mellon University
5000 Forbes Avenue
Pittsburgh, PA 15213 USA
{ mbilotti, ehn }@cs.cmu.edu

Abstract

Question Answering (QA) systems are often built modularly, with a text retrieval component feeding forward into an answer extraction component. Conventional wisdom suggests that, the higher the quality of the retrieval results used as input to the answer extraction module, the better the extracted answers, and hence system accuracy, will be. This turns out to be a poor assumption, because text retrieval and answer extraction are tightly coupled. Improvements in retrieval quality can be lost at the answer extraction module, which can not necessarily recognize the additional answer candidates provided by improved retrieval. Going forward, to improve accuracy on the QA task, systems will need greater coordination between text retrieval and answer extraction modules.

1 Introduction

The task of Question Answering (QA) involves taking a question phrased in natural human language and locating specific answers to that question expressed within a text collection. Regardless of system architecture, or whether the system is operating over a closed text collection or the web, most QA systems use text retrieval as a first step to narrow the search space for the answer to the question to a subset of the text collection (Hirschman and Gaizauskas, 2001). The remainder of the QA process amounts to a gradual narrowing of the search space, using successively more finely-grained filters to extract, validate and present one or more answers to the question.

Perhaps the most popular system architecture in the QA research community is the modular architecture, in most variations of which, text retrieval is represented as a separate component, isolated by a software abstraction from question analysis and answer extraction mechanisms. The widely-accepted pipelined modular architecture imposes a strict linear ordering on the system's control flow, with the analysis of the input question used as input to the text retrieval module, and the retrieved results feeding into the downstream answer extraction components.

Proponents of the modular architecture naturally view the QA task as decomposable, and to a certain extent, it is. The modules, however, can never be fully decoupled, because question analysis and answer extraction components, at least, depend on a common representation for answers and perhaps also a common set of text processing tools. This dependency is necessary to enable the answer extraction mechanism to determine whether answers exist in retrieved text, by analyzing it and comparing it against the question analysis module's answer specification. In practice, the text retrieval component does not use the common representation for scoring text; either the question analysis module or an explicit query formulation component maps it into a representation queryable by the text retrieval component.

The pipelined modular QA system architecture also carries with it an assumption about the compositionality of the components. It is easy to observe that errors cascade as the QA process moves through downstream modules, and this leads to the intuition that maximizing performance of individual modules minimizes the error at each stage of the pipeline, which, in turn, should maximize overall end-to-end system accuracy.

It is a good idea to pause to question what this intuition is telling us. Is end-to-end QA system performance really a linear function of individual

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 1–8
Manchester, UK. August 2008

[ARG0 [PERSON *John*]] [TARGET *loves*] [ARG1 [PERSON *Mary*]]

Figure 1: Example OpenEphyra semantic representation for the sentence, *John loves Mary*. Note that *John* is identified as the ARG0, the agent, or doer, of the *love* action. *Mary* is identified as the ARG1, the patient, or to whom the *love* action is being done. Both *John* and *Mary* are also identified as PERSON named entity types.

components? Is component performance really additive? This paper argues that the answer is no, not in general, and offers the counterexample of a high-precision text retrieval system that can check constraints against the common representation at retrieval time, which is integrated into a publicly-available pipelined modular QA system that is otherwise unchanged.

Ignoring the dependency between the answer extraction mechanism and the text retrieval component creates a problem. The answer extraction module is not able to handle the more sophisticated types of matches provided by the improved text retrieval module, and so it ignores them, leaving end-to-end system performance largely unchanged. The lesson learned is that a module improved in isolation does not necessarily provide an improvement in end-to-end system accuracy, and the paper concludes with recommendations for further research in bringing text retrieval and answer extraction closer together.

2 Improving Text Retrieval in Isolation

This section documents an attempt to improve the performance of a QA system by substituting its existing text retrieval component with for high-precision retrieval system capable of checking linguistic and semantic constraints at retrieval time.

2.1 The OpenEphyra QA System

OpenEphyra is the freely-available, open-source version of the Ephyra[1] QA system (Schlaefer et al., 2006; Schlaefer et al., 2007). OpenEphyra is a pipelined modular QA system having four stages: question analysis, query generation, search and answer extraction and selection. OpenEphyra also includes support for answer projection, or the use of the web to find answers to the question, which are then used to find supporting text in the corpus. Answer projection support was disabled for the purposes of this paper.

[1]See: `http://www.ephyra.info`

The common representation in OpenEphyra is a verb predicate-argument structure, augmented with named entity types, in which verb arguments are labeled with semantic roles in the style of PropBank (Kingsbury et al., 2002). This feature requires the separate download[2] of a semantic parser called ASSERT (Pradhan et al., 2004), which was trained on PropBank. See Figure 1 for an example representation for the sentence, *John loves Mary*.

OpenEphyra comes packaged with standard baseline methods for answer extraction and selection. For example, it extracts answers from retrieved text based on named entity instances matching the expected answer type as determined by the question analysis module. It can also look for predicate-argument structures that match the question structure, and can extract the argument corresponding to the argument in the question representing the interrogative phrase. OpenEphyra's default answer selection algorithm filters out answers containing question keyterms, merges subsets, and combines scores of duplicate answers.

2.2 Test Collection

The corpus used in this experiment is the AQUAINT corpus (Graff, 2002), the standard corpus for the TREC[3] QA evaluations held in 2002 through 2005. The corpus was prepared using MXTerminator (Reynar and Ratnaparkhi, 1997) for sentence segmentation, BBN Identifinder (Bikel et al., 1999) for named entity recognition, as well as the aforementioned ASSERT for identification of verb predicate-argument structures and PropBank-style semantic role labeling of the arguments.

The test collection consists of 109 questions from the QA track at TREC 2002 with extensive document-level relevance judgments (Bilotti et al., 2004; Lin and Katz, 2006) over the AQUAINT corpus. A set of sentence-level judgments was pre-

[2]See: `http://www.cemantix.org`

[3]Text REtrieval Conferences organized by the U.S. National Institute of Standards and Technology

Existing query	`#combine[sentence]( #any:person first person reach south pole )`
Top-ranked result	*Dufek became the first person to land an airplane at the South Pole.*
Second-ranked result	*He reached the North Pole in 1991.*
High-precision query	`#combine[sentence]( #max( #combine[target]( scored #max( #combine[./arg1]( #any:person )) #max( #combine[./arg2]( #max( #combine[target]( reach #max( #combine[./arg1]( south pole ))))))))`
Top-ranked result (relevant)	[ARG1 *Norwegian explorer* [PERSON *Roald Admundsen*]] [TARGET *becomes*] [ARG2 [ARG0 *first man*] *to* [TARGET *reach*] [ARG1 [LOCATION *South Pole*]]]

Figure 2: Retrieval comparison between OpenEphrya's existing text retrieval component, and the high-precision version it was a replaced with, for question 1475, *Who was the first person to reach the South Pole?* Note that the top two results retrieved by the existing text retrieval component are not relevant, and the top result from the high-precision component is relevant. The existing component does retrieve this answer-bearing sentence, but ranks it third.

pared by manually determining whether each sentence matching the TREC-provided answer pattern for a given question was *answer-bearing* according to the definition that an answer-bearing sentence completely contains and supports the answer to the question, without requiring inference or aggregation outside of that sentence. Questions without any answer-bearing sentences were removed from the test collection, leaving 91 questions.

Questions were manually reformulated so that they contain predicates. For example, question 1432, *Where is Devil's Tower?* was changed to *Where is Devil's Tower located?*, because ASSERT does not cover verbs, including *be* and *have*, that do not occur in its training data. Hand-corrected ASSERT parses for each question were were cached in the question analysis module. Reformulated questions are used as input to both the existing and high-precision text retrieval modules, to avoid advantaging one system over the other.

2.3 High-Precision Text Retrieval

OpenEphyra's existing text retrieval module was replaced with a high-precision text retrieval system based on a locally-modified version of the Indri (Strohman et al., 2005) search engine, a part of the open-source Lemur toolkit[4]. While the existing version of the text retrieval component supports querying on keyterms, phrases and placeholders for named entity types, the high-precision version also supports retrieval-time constraint-checking against the semantic representation based on verb predicate-argument structures, PropBank-style semantic role labels, and named entity recognition.

To make use of this expanded text retrieval capability, OpenEphyra's query formulation module was changed to source pre-prepared Indri queries that encode using structured query operators the predicate-argument and named entity constraints that match the answer-bearing sentences for each question. If questions have multiple queries associated with them, each query is evaluated individually, with the resulting ranked lists fused by Round Robin (Voorhees et al., 1994). Round Robin, which merges ranked lists by taking the top-ranked element from each list in order followed by lower-ranking elements, was chosen because Indri, the underlying retrieval engine, gives different queries scores that are not comparable in general, making it difficult to choose a fusion method that uses retrieval engine score as a feature.

Figure 2 shows a comparison of querying and retrieval behavior between OpenEphyra's existing text retrieval module and the high-precision version with which it is being replaced for question 1475, *Who was the first person to reach the South Pole?* The bottom of the figure shows an answer-bearing sentence with the correct answer, *Roald Admundsen*. The predicate-argument structure, se-

[4]See: `http://www.lemurproject.org`

mantic role labels and named entities are shown.

The high-precision text retrieval module supports storing of extents representing sentences, target verbs and arguments and named entity types as fields in the index. At query time, constraints on these fields can be checked using structured query operators. The queries in Figure 2 are shown in Indri syntax. Both queries begin with `#combine[sentence]`, which instructs Indri to score and rank sentence extents, rather than entire documents. The query for the existing text retrieval component contains keyterms as well an `#any:`*type* operator that matches instances of the expected answer type, which in this case is *person*. The high-precision query encodes a verb predicate-argument structure. The nested `#combine[target]` operator scores a sentence by the predicate-argument structures it contains. The `#combine[./`*role*`]` operators are used to indicate constraints on specific argument roles. The dot-slash syntax tells Indri that the argument extents are related to but not enclosed by the target extent. Throughout, the `#max` operator is used to select the best matching extent in the event that more than one satisfy the constraints.

Figure 3 compares average precision at the top twenty ranks over the entire question set between OpenEphyra's existing text retrieval module and the high-precision text retrieval module, showing that the latter performs better.

2.4 Results

To determine what effect improving text retrieval quality has on the end-to-end QA system, it suffices to run the system on the entire test collection,

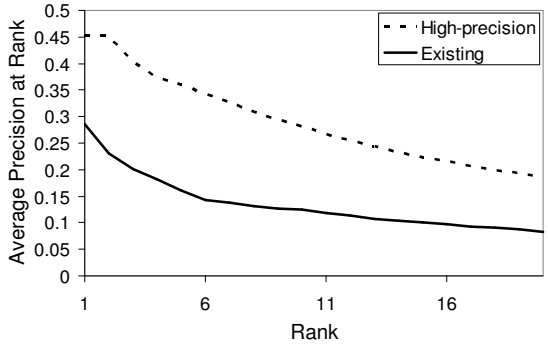

Figure 3: Comparison of average precision at top twenty ranks between OpenEphyra's existing text retrieval module, and the high-precision version that took its place.

replace the text retrieval component with the high-precision version while holding the other modules constant, and repeat the test run. Table 1 summarizes the MAP, average end-to-end system accuracy (whether the top-ranked returned answer is correct), and the mean reciprocal rank (MRR) of the correct answer (one over the rank at which the correct answer is returned). If the correct answer to a question is returned beyond rank twenty, the reciprocal rank for that question is considered to be zero.

Table 1: Summary of end-to-end QA system accuracy and MRR when the existing text retrieval module is replaced with a high-precision version

Retrieval	MAP	Accuracy	MRR
Existing	0.3234	0.1099	0.2080
High-precision	0.5487	0.1319	0.2020

Table 1 shows that, despite the improvement in average precision, the end-to-end system did not realize a significant improvement in accuracy or MRR. Viewed in the aggregate, the results are discouraging, because it seems that the performance gains realized after the text retrieval stage of the pipeline are lost in downstream answer extraction components.

Figure 4 compares OpenEphyra both before and after the integration of the high-precision text retrieval component on the basis of average precision and answer MRR. The horizontal axis plots the difference in average precision; a value of positive one indicates that the high-precision version of the module was perfect, ranking all answer-bearing sentences at the top of the ranked list, and that the existing version retrieved no relevant text at all. Negative one indicates the reverse. The vertical axis plots the difference in answer MRR. As before, positive one indicates that the high-precision component led the system to rank the correct answer first, and the existing component did not, and negative one indicates the reverse. The zero point on each axis is where the high-precision and existing text retrieval components performed equally well.

The expectation is that there will be a positive correlation between average precision and answer MRR; when the retrieval component provides higher quality results, the job of the answer extraction module should be easier. This is illustrated in the bottom portion of Figure 4, which was cre-

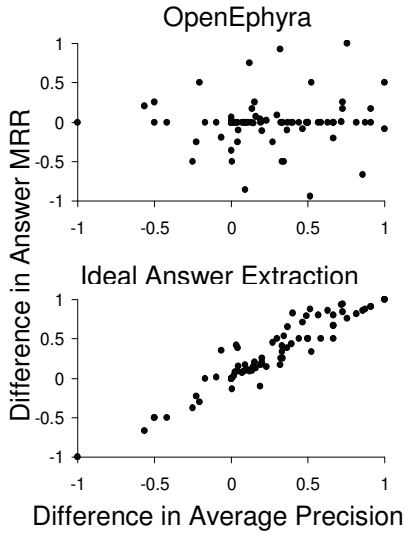

Figure 4: Scatter plot comparing the difference in average precision between the high-precision retrieval component and the existing retrieval component on the horizontal axis, to the difference in answer MRR on the vertical axis. Ideally, there would be a high correlation between the two; as average precision improves, so should answer MRR.

ated by assuming that the answer extraction module could successfully extract answers without error from all answer-bearing sentences returned by the text retrieval component.

Interestingly, actual extraction performance, shown in the top portion of Figure 4, bears little resemblance to the ideal. Note the large concentration of data points along the line representing zero difference in answer MRR. This indicates that, regardless of improvement in average precision of the results coming out of the retrieval module, the downstream answer extraction performance remains the same as it was when the existing text retrieval component was in use. This occurs because the answer extraction module does not know how to extract answers from some of the types of answer-bearing sentences retrieved by the high-precision version of the retrieval module and not by the existing version.

There are several data points in the top right-hand quadrant of the top half of Figure 4, indicating that for some questions, answer extraction was able to improve as average precision improved. This is likely due to better rankings for types of answer-bearing sentences that answer extraction already knows how to handle. Data points occurring in the lower right-hand portion of the graph indicate depressed answer extraction performance as average precision is increasing. This phenomenon can be explained by the higher-precision text retrieval module ranking answer-bearing sentences that answer extraction can not handle ahead of those that it can handle.

3 Failure Analysis

The results presented in the previous section confirm that an improvement made to the text retrieval component, in isolation, without a corresponding improvement to the downstream answer extraction modules, can fail to translate into a corresponding improvement in end-to-end QA system accuracy. The increased average precision in the retrieved results is coming in the form of answer-bearing sentences of types that the answer extraction machinery does not know how to handle. To address this gap in answer extraction coverage, it is first necessary to examine examples of the types of errors made by the OpenEphyra answer extraction module, summarized in Table 2.

Question 1497, *What was the original name before "The Star Spangled Banner"?* is an example of a question for which OpenEphyra's answer extraction machinery failed outright. An answer-bearing sentence was retrieved, however, containing the answer inside a quoted phrase: *His poem was titled "Defense of Fort M'Henry" and by November 1814 had been published as "The Star-Spangled Banner"*. The expected answer type of this question does not match a commonly-used named entity type, so OpenEphrya's named entity-based answer extractor found no candidates in this sentence. Predicate-argument structure-based answer extraction fails as well because the old and new names do not appear within the same structure. Because OpenEphyra does not include support for positing quoted phrases as answer candidates, no answer to this question can be found despite the fact that an answer-bearing sentence was retrieved.

Question 1417, *Who was the first person to run the mile in less than four minutes?* is an example of a question for which average precision improved greatly, by 0.7208, but for which extraction quality remained the same. The existing text retrieval module ranks 14 sentences ahead of the first answer-bearing sentence, but only one contains a named entity of type person, so despite the improvement in retrieval quality, the correct answer

Table 2: Summary of end-to-end QA system results on the question set

Result Type	Count
Extraction failure	42
Retrieval better, extraction same	20
Retrieval better, extraction worse	13
Retrieval better, extraction better	10
Retrieval worse, extraction better	3
Retrieval worse, extraction worse	3
Total	91

moves up only one rank in the system output.

For ten questions, extraction performance does improve as average precision improves. Question 1409, *Which vintage rock and roll singer was known as "The Killer"?* For each of these questions, OpenEphyra's existing text retrieval module could not rank an answer-bearing sentence highly or retrieve one at all. Adding the high-precision version of the text retrieval component solved this problem. In each case, named entity-based answer extraction was able extract the correct answer. These eleven questions range over a variety of answer types, and have little in common except for the fact that there are relatively few answer-bearing sentences in the corpus, and large numbers of documents matched by a bag-of-words query formulated using the keyterms from the question.

There are three questions for which extraction performance degrades as retrieval performance degrades. Question 1463, *What is the North Korean national anthem?* is an example. In this case, there is only one relevant sentence, and, owing to an annotation error, it has a predicate-argument structure that is very generic, having *North Korea* as the only argument: *Some of the North Korean coaches broke into tears as the North's anthem, the Patriotic Song, played.* The high-precision retrieval component retrieved a large number of sentences matching the that predicate-argument structure, but ranked the one answer-bearing sentence very low.

Some questions actually worsened in terms of the reciprocal rank of the correct answer when average precision improved. An example is question 1504, *Where is the Salton Sea?* The high-precision text retrieval module ranked answer-bearing sentences such as *The combination could go a long way to removing much of the pesticides, fertilizers, raw sewage carried by the river into the Salton Sea, the largest lake in California*, but a failure of the named entity recognition tool did not identify *California* as an instance of the expected answer type, and therefore it was ignored. Sentences describing other seas near other locations provided answers such as *Central Asia*, *Russia*, *Turkey* and *Ukraine* that were ranked ahead of *California*, which was eventually extracted from another answer-bearing sentence.

And finally, for some questions, high-precision retrieval was more of a hindrance than a help, retrieving more noise than answer-bearing sentences. A question for which this is true is question 1470, *When did president Herbert Hoover die?* The high-precision text retrieval module uses a predicate-argument structure to match the target verb *die*, theme *Hoover* and a *date* instance occurring in a temporal adjunct. Interestingly, the text collection contains a great deal of *die* structures that match partially, including those referring to deaths of presidents of other nations, and those referring to the death of J. Edgar Hoover, who was not a U.S. president but the first director of the U.S. Federal Bureau of Investigation (FBI). False positives such as these serve to push the true answer down on the ranked list of answers coming out of the QA system.

4 Improving Answer Extraction

The answer extraction and selection algorithms packaged with OpenEphyra are widely-accepted baselines, but are not sophisticated enough to extract answer candidates from the additional answer-bearing text retrieved by the high-precision text retrieval module, which can check linguistic and semantic constraints at query time.

The named-entity answer extraction method selects any candidate answer that is an instance of the expected answer type, so long as it co-occur with query terms. Consider question 1467, *What*

year did South Dakota become a state? Given that the corpus consists of newswire text reporting on current events, years that are contemporary to the corpus often co-occur with the question focus, as in the following sentence, *Monaghan also seized about $87,000 from a Santee account in South Dakota in 1997*. Of the top twenty answers returned for this question, all but four are contemporary to the corpus or in the future. Minimal sanity-checking on candidate answers could save the system the embarrassment of returning a date in the future as the answer. Going one step further would involve using external sources to determine that *1997* is too recent to be the year a state was admitted to the union.

OpenEphyra's predicate-argument structure-based answer extraction algorithm can avoid some of these noisy answers by comparing some constraints from the question against the retrieved text and only extracting answers if the constraints are satisfied. Consider question 1493, *When was Davy Crockett born?* One relevant sentence says *Crockett was born Aug. 17, 1786, in what is now eastern Tennessee, and moved to Lawrenceburg in 1817*. The SRL answer extraction algorithm extracts *Aug. 17, 1786* because it is located in an argument labeled *argm-tmp* with respect to the verb, and ignores the other date in the sentence, *1817*. The named entity-based answer extraction approach proposes both dates as answer candidates, but the redundancy-based answer selection prefers *1786*.

The predicate-argument structure-based answer extraction algorithm is limited because it only extracts arguments from text that shares the structure as the question. The high-precision text retrieval approach is actually able to retrieve additional answer-bearing sentences with different predicate-argument structures from the question, but answer extraction is not able to make use of it. Consider the sentence, *At the time of his 100 point game with the Philadelphia Warriors in 1962, Chamberlain was renting an apartment in New York*. Though this sentence answers the question *What year did Wilt Chamberlain score 100 points?*, its predicate-argument structure is different from that of the question, and predicate-argument structure-based answer extraction will ignore this result because it does not contain a *score* verb.

In addition to answer extraction, end-to-end performance could be improved by focusing on answer selection. OpenEphyra does not include support for sanity-checking the answers it returns, and its default answer selection mechanism is redundancy-based. As a result, nonsensical answers are occasionally retrieved, such as *moon* for question 1474, *What is the lowest point on Earth?* Sophisticated approaches, however, do exist for answer validation and justification, including use of resources such as gazetteers and ontologies (Buscaldi and Rosso, 2006), Wikipedia (Xu et al., 2002), the Web (Magnini et al., 2002), and combinations of the above (Ko et al., 2007).

5 Conclusions

This paper set out to challenge the assumption of compositionality in pipelined modular QA systems that suggests that an improvement in an individual module should lead to an improvement in the overall end-to-end system performance. An attempt was made to validate the assumption by showing an improvement in the end-to-end system accuracy of an off-the-shelf QA system by substituting its existing text retrieval component for a high-precision retrieval component capable of checking linguistic and semantic constraints at query time. End-to-end system accuracy remained roughly unchanged because the downstream answer extraction components were not able to extract answers from the types of the answer-bearing sentences returned by the improved retrieval module.

The reality of QA systems is that there is a high level of coupling between the different system components. Ideally, text retrieval should have an understanding of the kinds of results that answer extraction is able to utilize to extract answers, and should not offer text beyond the capabilities of the downstream modules. Similarly, question analysis and answer extraction should be agreeing on a common representation for what constitutes an answer to the question so that answer extraction can use that information to locate answers in retrieved text. When a retrieval module is available that is capable of making use of the semantic representation of the answer, it should do so, but answer extraction needs to know what it can assume about incoming results so that it does not have to re-check constraints already guaranteed to hold.

The coupling between text retrieval and answer extraction is important for a QA system to perform well. Improving the quality of text retrieval is essential because once the likely location of

the answer is narrowed down to a subset of the text collection, anything not retrieved text can not be searched for answers in downstream modules. Equally important is the role of answer extraction. Even the most relevant retrieved text is useless to a QA system unless answers can be extracted from it. End-to-end QA system performance can not be improved by improving text retrieval quality in isolation. Improvements in answer extraction must keep pace with progress on text retrieval techniques to reduce errors resulting from a mismatch in capabilities. Going forward, research on the linguistic and semantic constraint-checking capabilities of text retrieval systems to support the QA task can drive research in answer extraction techniques, and in QA systems in general.

References

Bikel, D., R. Schwartz, and R. Weischedel. 1999. An algorithm that learns what's in a name. *Machine Learning*, 34(1–3):211–231.

Bilotti, M., B. Katz, and J. Lin. 2004. What works better for question answering: Stemming or morphological query expansion? In *Proceedings of the Information Retrieval for Question Answering (IR4QA) Workshop at SIGIR 2004*.

Bilotti, M., P. Ogilvie, J. Callan, and E. Nyberg. 2007. Structured retrieval for question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Buscaldi, D. and P. Rosso. 2006. Mining knowledge from wikipedia for the question answering task. In *Proceedings of the International Conference on Language Resources and Evaluation*.

Cui, H., R. Sun, K. Li, M. Kan, and T. Chua. 2005. Question answering passage retrieval using dependency relations. In *Proceedings of the 28th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Graff, D. 2002. *The AQUAINT Corpus of English News Text*. Linguistic Data Consortium (LDC). Cat. No. LDC2002T31.

Hirschman, L. and R. Gaizauskas. 2001. Natural language question answering: The view from here. *Journal of Natural Language Engineering, Special Issue on Question Answering*, Fall–Winter.

Kingsbury, P., M. Palmer, and M. Marcus. 2002. Adding semantic annotation to the penn treebank. In *Proceedings of the 2nd International Conference on Human Language Technology Research (HLT 2002)*.

Ko, J., L. Si, and E. Nyberg. 2007. A probabilistic graphical model for joint answer ranking in question answering. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Lin, J. and B. Katz. 2006. Building a reusable test collection for question answering. *Journal of the American Society for Information Science and Technology*, 57(7):851–861.

Magnini, B., M. Negri, R. Pervete, and H. Tanev. 2002. Comparing statistical and content-based techniques for answer validation on the web. In *Proceedings of the VIIIo Convegno AI*IA*.

Narayanan, S. and S. Harabagiu. 2004. Question answering based on semantic structures. In *Proceedings of the 20th international conference on Computational Linguistics*.

Pradhan, S., W. Ward, K. Hacioglu, J. Martin, and D. Jurafsky. 2004. Shallow semantic parsing using support vector machines. In *Proceedings of the Human Language Technology Conference of the North American Chapter of the Association for Computational Linguistics (HLT-NAACL 2004)*.

Reynar, J. and A. Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proceedings of the Fifth Conference on Applied Natural Language Processing*.

Schlaefer, N., P. Gieselmann, and G. Sautter. 2006. The ephyra qa system at trec 2006. In *Proceedings of the Fifteenth Text REtrieval Conference (TREC)*.

Schlaefer, N., J. Ko, J. Betteridge, G. Sautter, M. Pathak, and E. Nyberg. 2007. Semantic extensions of the ephyra qa system for trec 2007. In *Proceedings of the Sixteenth Text REtrieval Conference (TREC)*.

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

Sun, R., J. Jiang, Y. Tan, H. Cui, T. Chua, and M. Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *Proceedings of the Fourteenth Text REtrieval Conference (TREC-14)*.

Voorhees, E., N. Gupta, and B. Johnson-Laird. 1994. The collection fusion problem. In *Proc. of TREC-3*.

Xu, J., A. Licuanan, J. May, S. Miller, and R. Weischedel. 2002. Trec 2002 qa at bbn: Answer selection and confidence estimation. In *Proceedings of the Text REtrieval Conference*.

Exact Phrases in Information Retrieval for Question Answering

Svetlana Stoyanchev, and Young Chol Song, and William Lahti
Department of Computer Science
Stony Brook University
Stony Brook, NY 11794-4400
svetastenchikova, nskystars, william.lahti @gmail.com

Abstract

Question answering (QA) is the task of finding a concise answer to a natural language question. The first stage of QA involves information retrieval. Therefore, performance of an information retrieval subsystem serves as an upper bound for the performance of a QA system. In this work we use phrases automatically identified from questions as exact match constituents to search queries. Our results show an improvement over baseline on several document and sentence retrieval measures on the WEB dataset. We get a 20% relative improvement in MRR for sentence extraction on the WEB dataset when using automatically generated phrases and a further 9.5% relative improvement when using manually annotated phrases. Surprisingly, a separate experiment on the indexed AQUAINT dataset showed no effect on IR performance of using exact phrases.

1 Introduction

Question answering can be viewed as a sophisticated information retrieval (IR) task where a system automatically generates a search query from a natural language question and finds a concise answer from a set of documents. In the open-domain factoid question answering task systems answer general questions like *Who is the creator of The Daily Show?*, or *When was Mozart born?*. A variety of approaches to question answering have been investigated in TREC competitions in the last decade from (Vorhees and Harman, 1999) to (Dang et al., 2006). Most existing question answering systems add question analysis, sentence retrieval and answer extraction components to an IR system.

Since information retrieval is the first stage of question answering, its performance is an upper bound on the overall question answering system's performance. IR performance depends on the quality of document indexing and query construction. Question answering systems create a search query automatically from a user's question, through various levels of sophistication. The simplest way of creating a query is to treat the words in the question as the terms in the query. Some question answering systems (Srihari and Li, 1999) apply linguistic processing to the question, identifying named entities and other query-relevant phrases. Others (Hovy et al., 2001b) use ontologies to expand query terms with synonyms and hypernyms.

IR system recall is very important for question answering. If no correct answers are present in a document, no further processing will be able to find an answer. IR system precision and ranking of candidate passages can also affect question answering performance. If a sentence without a correct answer is ranked highly, answer extraction may extract incorrect answers from these erroneous candidates. Collins-Thompson *et al.* (2004) show that there is a consistent relationship between the quality of document retrieval and the overall performance of question answering systems.

In this work we evaluate the use of *exact phrases* from a question in document and passage retrieval. First, we analyze how different parts of a question contribute to the performance of the sentence extraction stage of question answering. We ana-

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 9–16
Manchester, UK. August 2008

lyze the match between linguistic constituents of different types in questions and sentences containing candidate answers. For this analysis, we use a set of questions and answers from the TREC 2006 competition as a *gold standard*.

Second, we evaluate the performance of document retrieval in our *StoQA* question answering system. We compare the performance of document retrieval from the Web and from an indexed collection of documents using different methods of query construction, and identify the optimal algorithm for query construction in our system as well as its limitations.

Third, we evaluate passage extraction from a set of documents. We analyze how the specificity of a query affects sentence extraction.

The rest of the paper is organized as follows: In Section 2, we summarize recent approaches to question answering. In Section 3, we describe the dataset used in this experiment. In Section 5, we describe our method and data analysis. In Section 4, we outline the architecture of our question answering system. In Section 6, we describe our experiments and present our results. We summarize in Section 7.

2 Related Work

Information retrieval (IR) for question answering consists of 2 steps: document retrieval and passage retrieval.

Approaches to passage retrieval include simple word overlap (Light et al., 2001), density-based passage retrieval (Clarke et al., 2000), retrieval based on the inverse document frequency (IDF) of matched and mismatched words (Ittycheriah et al., 2001), cosine similarity between a question and a passage (Llopis and Vicedo, 2001), passage/sentence ranking by weighting different features (Lee and others, 2001), stemming and morphological query expansion (2004), and voting between different retrieval methods (Tellex et al., 2003). As in previous approaches, we use words and phrases from a question for passage extraction and experiment with using exactly matched phrases in addition to words. Similarly to Lee (2001), we assign weights to sentences in retrieved documents according to the number of matched constituents.

Systems vary in the size of retrieved passages. Some systems identify multi-sentence and variable size passages (Ittycheriah et al., 2001; Clarke et al., 2000). An optimal passage size may depend on the method of answer extraction. We use single sentence extraction because our system's semantic role labeling-based answer extraction functions on individual sentences.

White and Sutcliffe (2004) performed a manual analysis of questions and answers for 50 of the TREC questions. The authors computed frequency of terms matching exactly, with morphological, or semantic variation between a question and a answer passage. In this work we perform a similar analysis automatically. We compare frequencies of phrases and words matching between a question and candidate sentences.

Query expansion has been investigated in systems described in (Hovy et al., 2001a; Harabagiu et al., 2006). They use WordNet (Miller, 1995) for query expansion, and incorporate semantic roles in the answer extraction process. In this experiment we do not expand query terms.

Corpus pre-processing and encoding information useful for retrieval was shown to improve document retrieval (Katz and Lin, 2003; Harabagiu et al., 2006; Chu-Carroll et al., 2006). In our approach we evaluate linguistic question processing technique which does not require corpus preprocessing.

Statistical machine translation model is used for information retrieval by (Murdock and Croft, 2005). The model estimates probability of a question given an answer and is trained on <question, candidate sentence> pairs. It capturing synonymy and grammar transformations using a statistical model.

3 Data

In this work we evaluate our question answering system on two datasets: the AQUAINT corpus, a 3 gigabyte collection of news documents used in the TREC 2006 competition; and the Web.

We use questions from TREC, a yearly question answering competition. We use a subset of questions with non-empty answers [1] from the TREC 2006 dataset [2]. The dataset provides a list of matching documents from the AQUAINT corpus and correct answers for each question. The dataset contains 387 questions; the AQUAINT corpus contains an average of 3.5 documents per ques-

[1] The questions where an answer was not in the dataset were not used in this analysis

[2] http://trec.nist.gov/data/qa/t2006_qadata.html

tion that contain the correct answer to that question. Using *correct answers* we find the *correct sentences* from the *matching documents*. We use this information as a gold standard for the IR task.

We index the documents in the AQUAINT corpus using the Lucene (Apache, 2004 2008) engine on the document level. We evaluate document retrieval using *gold standard* documents from the AQUAINT corpus. We evaluate sentence extraction on both AQUAINT and the Web automatically using regular expressions for correct answers provided by TREC.

In our experiments we use manually and automatically created phrases. Our automatically created phrases were obtained by extracting noun, verb and prepositional phrases and named entities from the question dataset using then NLTK (Bird et al., 2008) and Lingpipe (Carpenter and Baldwin, 2008) tools. Our manually created phrases were obtained by hand-correcting these automatic annotations (e.g. to remove extraneous words and phrases and add missed words and phrases from the questions).

4 System

For the experiments in this paper we use the *StoQA* system. This system employs a pipeline architecture with three main stages as illustrated in Figure 1: question analysis, document and sentence extraction (IR), and answer extraction. After the user poses a question, it is analyzed. Target named entities and semantic roles are determined. A query is constructed, tailored to the search tools in use. Sentences containing target terms are then extracted from the documents retrieved by the query. The candidate sentences are processed to identify and extract candidate answers, which are presented to the user.

We use the NLTK toolkit (Bird et al., 2008) for question analysis and can add terms to search queries using WordNet (Miller, 1995). Our system can currently retrieve documents from either the Web (using the Yahoo search API (Yahoo!, 2008)), or the AQUAINT corpus (Graff, 2002) (through the Lucene indexer and search engine (Apache, 2004 2008)). When using Lucene, we can assign different weights to different types of search term (e.g. less weight to terms than to named entities added to a query) (cf. (Lee and others, 2001)).

We currently have two modules for answer extraction, which can be used separately or together. Candidate sentences can be tagged with named entity information using the Lydia system (Lloyd et al., 2005). The tagged word/phrase matching the target named entity type most frequently found is chosen as the answer. Our system can also extract answers through semantic role labeling, using the SRL toolkit from (Punyakanok et al., 2008). In this case, the tagged word/phrase matching the target semantic role most frequently found is chosen as the answer.

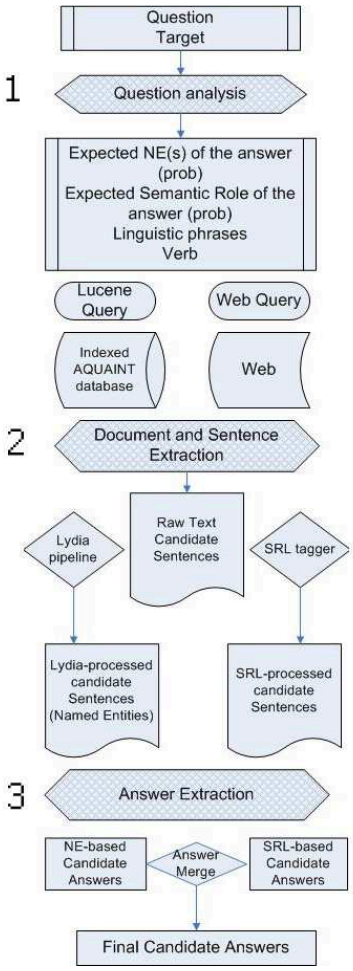

Figure 1: Architecutre of our question answering system

5 Method

5.1 Motivation

Question answering is an engineering-intensive task. System performance improves as more sophisticated techniques are applied to data processing. For example, the IR stage in question answering is shown to improve with the help of techniques like predictive annotations and relation extraction; matching of semantic and syntactic re-

Target Question	United Nations What was the number of member nations of the U.N. in 2000?
Named Entity	U.N., United Nations
Phrases	"member nations of the U.N."
Converted Q-phrase	"member nations of the U.N. in 2000"
Baseline Query	was the number of member nations of the U.N. in 2000 United Nations
Lucene Query with phrases and NE	was the number of member nations of the U.N. in 2000 "United Nations", "member nations of the u.n."
Cascaded web query	
query1	"member nations of the U.N. in 2000" AND (United Nations)
query2	"member nations of the u.n." AND (United Nations)
query3	(number of member nations of the U.N. in 2000) AND (United Nations)
query4	(United Nations)

Table 1: Question processing example: terms of a query

lations in a question and a candidate sentence are known to improve overall QA system performance (Prager et al., 2000; Stenchikova et al., 2006; Katz and Lin, 2003; Harabagiu et al., 2006; Chu-Carroll et al., 2006).

In this work we analyze less resource expensive techniques, such as chunking and named entity detection, for IR in question answering. Linguistic analysis in our system is applied to questions and to candidate sentences only. There is no need for annotation of all documents to be indexed, so our techniques can be applied to IR on large datasets such as the Web.

Intuitively, using phrases in query construction may improve retrieval precision. For example, if we search for *In what year did the movie win academy awards?* using a disjunction of words as our query we may match irrelevant documents about the military *academy* or Nobel prize *awards*. However, if we use the phrase *"academy awards"* as one of the query terms, documents with this term will receive a higher ranking. A counterargument for using phrases is that *academy* and *awards* are highly correlated and therefore the documents that contain both will be more highly ranked. We hypothesize that for phrases where constituents are not highly correlated, exact phrase extraction will give more benefit.

5.2 Search Query

We process each TREC question and target [3] to identify named entities. Often, the target is a complete named entity (NE), however, in some of the TREC questions the target contains a named entity, e.g. *tourists massacred at Luxor in 1997*, or *1991 eruption of Mount Pinatubo* with named entities *Luxor* and *Mount Pinatubo*. For the TREC question *What was the number of member nations of the U.N. in 2000?*, the identified constituents and automatically constructed query are shown in Table 1. **Named entities** are identified using Lingpipe (Carpenter and Baldwin, 2008), which identifies named entities of type *organization, location* and *person*. **Phrases** are identified automatically using the NLTK toolkit (Bird et al., 2008). We extract noun phrases, verb phrases and prepositional phrases. The rules for identifying phrases are mined from a dataset of manually annotated parse trees (Judge et al., 2006) [4]. **Converted Q-phrases** are heuristically created phrases that paraphrase the question in declarative form using a small set of rules. The rules match a question to a pattern and transform the question using linguistic information. For example, one rule matches *Who is|was NOUN|PRONOUN VBD* and converts it to *NOUN|PRONOUN is|was VBD*. [5]

[3] The TREC dataset also provides a *target topic* for each questions, and we include it in the query.

[4] The test questions are not in this dataset.

[5] Q-phrase is extracted only for who/when/where questions. We used a set of 6 transformation patterns in this experiment.

Named Entities	Phrases
great pyramids; frank sinatra; mt. pinatubo; miss america; manchester united; clinton administration	capacity of the ballpark; groath rate; security council; tufts university endowment; family members; terrorist organization

Table 2: Automatically identified named entities and phrases

A q-phrase represents how a simple answer is expected to appear, e. g. a **q-phrase** for the question *When was Mozart born?* is *Mozart was born.* We expect a low probability of encountering a **q-phrase** in retrieved documents, but a high probability of co-occurrence of q-phrases phrase with correct answers.

In our basic system (baseline), words (trivial query constituents) from question and target form the query. In the experimental system, the query is created from a combination of words, quoted exact phrases, and quoted named entities. Table 2 shows some examples of phrases and named entities used in queries. The goal of our analysis is to evaluate whether non-trivial query constituents can improve document and sentence extraction.

We use a back-off mechanism with both of our IR subsystems to improve document extraction. The Lucene API allows the user to create arbitrarily long queries and assign a weight to each query constituent. We experiment with assigning different weights based on the type of a query constituent. Assigning a higher weight to phrase constituents increases the scores for documents matching a phrase, but if no phrase matches are found documents matching lower-scored constituents will be returned.

The query construction system for the Web first produces a query containing only **converted q-phrases** which have low recall and high precision (query 1 in table 1). If this query returns less than 20 results, it then constructs a query using **phrases** (query 2 in table 1), if this returns less than 20 results, queries without exact phrases (queries 3 and 4) are used. Every query contains a conjunction with the question *target* to increase precision for the cases where the *target* is excluded from **converted q-phrase** or an **exact phrase**.

For both our IR subsystems we return a maximum of 20 documents. We chose this relatively low number of documents because our answer extraction algorithm relies on semantic tagging of candidate sentences, which is a relatively time-consuming operation.

The text from each retrieved documents is split into sentences using Lingpipe. The same sentence extraction algorithm is used for the output from both IR subsystems (AQUAINT/Lucene and Web/Yahoo). The sentence extraction algorithm assigns a score to each sentence according to the number of matched terms it contains.

5.3 Analysis of Constituents

For our analysis of the impact of different linguistic constituent types on document retrieval we use the TREC 2006 dataset which consists of questions, documents containing answers to each question, and *supporting sentences*, sentences from these documents that contain the answer to each question.

Table 3 shows the number of times each constituent type appears in a *supporting sentence* and the proportion of *supporting sentences* containing each constituent type (sent w/answer column). The "All Sentences" column shows the number of constituents in all sentences of candidate documents. The *precision* column displays the chance that a given sentence is a *supporting sentence* if a constituent of a particular type is present in it. *Converted q-phrase* has the highest precision, followed by phrases, verbs, and named entities. Words have the highest chance of occurrence in a *supporting sentence* (.907), but they also have a high chance of occurrence in a document (.745).

This analysis supports our hypothesis that using exact phrases may improve the performance of information retrieval for question answering.

6 Experiment

In these experiments we look at the impact of using exact phrases on the performance of the document retrieval and sentence extraction stages of question answering. We use our *StoQA* question answering system. Questions are analyzed as described in the previous section. For document retrieval we use the back-off method described in the previous sec-

	sent w/ answer		all sentences		precision
	num	proportion	num	proportion	
Named Entity	907	0.320	4873	0.122	.18
Phrases	350	0.123	1072	0.027	.34
Verbs	396	0.140	1399	0.035	.28
Q-Phrases	11	0.004	15	0.00038	.73
Words	2573	0.907	29576	0.745	.086
Total Sentences	2836		39688		

Table 3: Query constituents in sentences of correct documents

	avg doc recall	avg doc MRR	overall doc recall	avg sent MRR	overall sent recall	avg corr sent in top 1	avg corr sent in top 10	avg corr sent in top 50
IR with Lucene on AQUAINT dataset								
baseline (words disjunction from target and question)	0.530	0.631	0.756	0.314	0.627	0.223	1.202	3.464
baseline + auto phrases	0.514	0.617	0.741	0.332	0.653	0.236	1.269	3.759
words + auto NEs & phrases	0.501	0.604	0.736	0.316	0.653	0.220	1.228	3.705
baseline + manual phrases	0.506	0.621	0.738	0.291	0.609	0.199	1.231	3.378
words + manual NEs & phrases	0.510	0.625	0.738	0.294	0.609	0.202	1.244	3.368
IR with Yahoo API on WEB								
baseline words disjunction	-	-	-	0.183	0.570	0.101	0.821	2.316
cascaded using auto phrases	-	-	-	0.220	0.604	0.140	0.956	2.725
cascaded using manual phrases	-	-	-	0.241	0.614	0.155	1.065	3.016

Table 4: Document retrieval evaluation.

tion. We performed the experiments using first automatically generated phrases, and then manually corrected phrases.

For document retrieval we report: 1) average recall, 2) average mean reciprocal ranking (MRR), and 3) overall document recall. Each question has a document retrieval recall score which is the proportion of documents identified from all correct documents for this question. The *average recall* is the individual recall averaged over all questions. MRR is the inverse index of the first correct document. For example, if the first correct document appears second, the MRR score will be 1/2. MRR is computed for each question and averaged over all questions. *Overall document recall* is the percentage of questions for which at least one correct document was retrieved. This measure indicates the upper bound on the QA system.

For sentence retrieval we report 1) average sentence MRR, 2) overall sentence recall, 3) average precision of the first sentence, 4) number of correct candidate sentences in the top 10 results, and 5) number of correct candidate sentences in the top 50 results [6].

Table 4 shows our experimental results. First, we evaluate the performance of document retrieval on the indexed AQUAINT dataset. Average document recall for our baseline system is 0.53, indicating that on average half of the correct documents are retrieved. Average document MRR is .631, meaning that on average the first correct document appears first or second. Overall document recall indicates that 75.6% of queries contain a correct document among the retrieved documents. Average sentence recall is lower than document recall indicating that some proportion of correct answers is not retrieved using our heuristic sentence extraction algorithm. The average sentence MRR is .314 indicating that the first correct sentence is approximately third on the list. With

[6] Although the number of documents is 20, multiple sentences may be extracted from each document.

the AQUAINT dataset, we notice no improvement with exact phrases.

Next, we evaluate sentence retrieval from the WEB. There is no *gold standard* for the WEB dataset so we do not report document retrieval scores. Sentence scores on the WEB dataset are lower than on the AQUAINT dataset [7].

Using back-off retrieval with automatically created phrases and named entities, we see an improvement over the baseline system performance for each of the sentence measures on the WEB dataset. Average sentence MRR increases 20% from .183 in the baseline to .220 in the experimental system. With manually created phrases MRR improves a further 9.5% to .241. This indicates that information retrieval on the WEB dataset can benefit from a better quality of chunker and from a properly converted question phrase. It also shows that the improvement is not due to simply matching random substrings from a question, but that linguistic information is useful in constructing the exact match phrases. Precision of automatically detected phrases is affected by errors during automatic part-of-speech tagging of questions. An example of an error due to POS tagging is the identification of a phrase *was Rowling born* due to a failure to identify that *born* is a *verb*.

Our results emphasize the difference between the two datasets. AQUAINT dataset is a collection of a large set of news documents, while WEB is a much larger resource of information from a variety of sources. It is reasonable to assume that on average there are much fewer documents with query words in AQUAINT corpus than on the WEB. Proportion of *correct documents* from all retrieved WEB documents on average is likely to be lower than this proportion in documents retrieved from AQUAINT. When using words on a query to AQUAINT dataset, most of the *correct documents* are returned in the top matches. Our results indicate that over 50% of *correct documents* are retrieved in the top 20 results. Results in table 3 indicate that exactly matched phrases from a question are more precise predictors of presence of an answer. Using exact matched phrases in a WEB query allows a search engine to give higher rank to more relevant documents and increases likelihood of these documents in the top 20 matches.

Although overall performance on the WEB dataset is lower than on AQUAINT, there is a potential for improvement by using a larger set of documents and improving our sentence extraction heuristics.

[7] Our decision to use only 20 documents may be a factor.

7 Conclusion and Future Work

In this paper we present a document retrieval experiment on a question answering system. We evaluate the use of named entities and of noun, verb, and prepositional phrases as exact match phrases in a document retrieval query. Our results indicate that using phrases extracted from questions improves IR performance on WEB data. Surprisingly, we find no positive effect of using phrases on a smaller closed set of data.

Our data analysis shows that linguistic phrases are more accurate indicators for candidate sentences than words. In future work we plan to evaluate how phrase type (noun vs. verb vs. preposition) affects IR performance.

Acknowledgment

We would like to thank professor Amanda Stent for suggestions about experiments and proofreading the paper. We would like to thank the reviewers for useful comments.

References

Apache. 2004-2008. Lucene. http://lucene.apache.org/java/docs/index.html.

Bilotti, M., B. Katz, and J. Lin. 2004. What works better for question answering: Stemming or morphological query expansion? In *Proc. SIGIR*.

Bird, S., E. Loper, and E. Klein. 2008. Natural Language ToolKit (NLTK). http://nltk.org/index.php/Main_Page.

Carpenter, B. and B. Baldwin. 2008. Lingpipe. http://alias-i.com/lingpipe/index.html.

Chu-Carroll, J., J. Prager, K. Czuba, D. Ferrucci, and P. Duboue. 2006. Semantic search via XML fragments: a high-precision approach to IR. In *Proc. SIGIR*.

Clarke, C., G. Cormack, D. Kisman, and T. Lynam. 2000. Question answering by passage selection (multitext experiments for TREC-9). In *Proc. TREC*.

Collins-Thompson, K., J. Callan, E. Terra, and C. L.A. Clarke. 2004. The effect of document retrieval quality on factoid question answering performance. In *Proc. SIGIR*.

Dang, H., J. Lin, and D. Kelly. 2006. Overview of the TREC 2006 question answering track. In *Proc. TREC*.

Graff, D. 2002. The AQUAINT corpus of English news text. Technical report, Linguistic Data Consortium, Philadelphia, PA, USA.

Harabagiu, S., A. Hickl, J. Williams, J. Bensley, K. Roberts, Y. Shi, and B. Rink. 2006. Question answering with LCC's CHAUCER at TREC 2006. In *Proc. TREC*.

Hovy, E., L. Gerber, U. Hermjakob, M. Junk, and C.-Y. Lin. 2001a. Question answering in Webclopedia. In *Proc. TREC*.

Hovy, E., U. Hermjakob, and C.-Y. Lin. 2001b. The use of external knowledge in factoid QA. In *Proc. TREC*.

Ittycheriah, A., M. Franz, and S. Roukos. 2001. IBM's statistical question answering system – TREC-10. In *Proc. TREC*.

Judge, J., A. Cahill, and J. van Genabith. 2006. QuestionBank: Creating a corpus of parse-annotated questions. In *Proc. ACL*.

Katz, B. and J. Lin. 2003. Selectively using relations to improve precision in question answering. In *Proc. of the EACL Workshop on Natural Language Processing for Question Answering*.

Lee, G. G. et al. 2001. SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP. In *Proc. TREC*.

Light, M., G. S. Mann, E. Riloff, and E. Breck. 2001. Analyses for elucidating current question answering technology. *Journal of Natural Language Engineering*, 7(4).

Llopis, F. and J. L. Vicedo. 2001. IR-n: A passage retrieval system at CLEF-2001. In *Proc. of the Second Workshop of the Cross-Language Evaluation Forum (CLEF 2001)*.

Lloyd, L., D. Kechagias, and S. Skiena. 2005. Lydia: A system for large-scale news analysis. In *Proc. SPIRE*, pages 161–166.

Miller, George A. 1995. WordNet: a lexical database for english. *Communications of the ACM*, 38(11).

Murdock, V. and W. B. Croft. 2005. Simple translation models for sentence retrieval in factoid question answering. In *Proc. SIGIR*.

Prager, J., E. Brown, and A. Coden. 2000. Question-answering by predictive annotation. In *ACM SIGIR*. QA -to site.

Punyakanok, V., D. Roth, and W. Yih. 2008. The importance of syntactic parsing and inference in semantic role labeling. *Computational Linguistics*, 34(2).

Srihari, R. and W. Li. 1999. Information extraction supported question answering. In *Proc. TREC*.

Stenchikova, S., D. Hakkani-Tur, and G. Tur. 2006. QASR: Question answering using semantic roles for speech interface. In *Proc. ICSLP-Interspeech 2006*.

Tellex, S., B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proc. SIGIR*.

Vorhees, V. and D. Harman. 1999. Overview of the eighth Text REtrieval Conference (TREC-8). In *"Proc. TREC"*.

White, K. and R. Sutcliffe. 2004. Seeking an upper bound to sentence level retrieval in question answering. In *Proc. SIGIR*.

Yahoo!, Inc. 2008. Yahoo! search API. http://developer.yahoo.com/search/.

Simple is Best: Experiments with Different Document Segmentation Strategies for Passage Retrieval

Jörg Tiedemann
Information Science
University of Groningen
j.tiedemann@rug.nl

Jori Mur
Information Science
University of Groningen
j.mur@rug.nl

Abstract

Passage retrieval is used in QA to filter large document collections in order to find text units relevant for answering given questions. In our QA system we apply standard IR techniques and index-time passaging in the retrieval component. In this paper we investigate several ways of dividing documents into passages. In particular we look at semantically motivated approaches (using coreference chains and discourse clues) compared with simple window-based techniques. We evaluate retrieval performance and the overall QA performance in order to study the impact of the different segmentation approaches. From our experiments we can conclude that the simple techniques using fixed-sized windows clearly outperform the semantically motivated approaches, which indicates that uniformity in size seems to be more important than semantic coherence in our setup.

1 Introduction

Passage retrieval in question answering is different from information retrieval in general. Extracting relevant passages from large document collections is only one step in answering a natural language question. There are two main differences: i) Passage retrieval queries are generated from complete sentences (questions) compared to bag-of-keyword queries usually used in IR. ii) Retrieved passages have to be processed further in order to extract concrete answers to the given question. Hence, the size of the passages retrieved is important and smaller units are preferred. Here, the division of documents into passages is crucial. The textual units have to be big enough to ensure IR works properly and they have to be small enough to enable efficient and accurate QA. In this study we investigate whether semantically motivated passages in the retrieval component lead to better QA performance compared to the use of document retrieval and window-based segmentation approaches.

1.1 Index-time versus Search-time Passaging

In this paper, we experiment with various possibilities of dividing documents into passages *before* indexing them. This is also called *index-time* passaging and refers to a one-step process of retrieving appropriate textual units for subsequent answer extraction modules (Roberts and Gaizauskas, 2004; Greenwood, 2004). This is in contrast to other strategies using a two-step procedure consisting of document retrieval and *search-time* passaging thereafter. Here, we can distinguish between approaches that only return one passage per relevant document (see, for example, (Robertson et al., 1992)) and the ones that allow multiple passages per document (see, for example (Moldovan et al., 2000)). In general, allowing multiple passages per document is preferable for QA as possible answers can be contained at various positions in a document (Roberts and Gaizauskas, 2004). For this, an index-time approach has the advantage that the retrieval of multiple passages per documents is straightforward because all of them compete which each other in the same index using the same metric for ranking.

A comparison between index-time and search-

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 17–25
Manchester, UK. August 2008

time passaging has been carried out in (Roberts and Gaizauskas, 2004). In their experiments, index-time passaging performs similarly to search-time passaging in terms of coverage and redundancy (measures which have been introduced in the same paper; see section 4.2 for more information). Significant differences between the various approaches can only be observed in redundancy on higher ranks (above 50). However, as we will see later in our experiments (section 4.2), redundancy is not as important as coverage for our QA system . Furthermore, retrieving more than about 40 passages does not produce significant improvements of the QA system anymore but slows down the processing time substantially.

Another argument for our focus on a one-step retrieval procedure can be taken from (Tellex et al., 2003). In this paper, the authors do not actually use any index-time passaging approach but compare various search-time passage retrieval algorithms. However, they obtain a huge performance difference when applying an oracle document retriever (only returning relevant documents in the first retrieval step) instead of a standard IR engine. Compared to this, the differences between the various passage retrieval approaches tested is very small. From this we can conclude that much improvement can be gained by improving the initial retrieval step, which seems to be the bottleneck in the entire process. Unfortunately, the authors do not compare their results with index-time approaches. However, looking at the potential gain in document retrieval and keeping in mind that the performance of index-time and search-time approaches is rather similar (as we have discussed earlier) we believe that the index-time approach is preferable.

1.2 Passages in IR

Certainly, IR performance is effected by changing the size of the units to be indexed. The task in document segmentation for our index-time passaging approach is to find the proper division of documents into text passages which optimize the retrieval in terms of overall QA performance.

The general advantages of passage retrieval over full-text document retrieval has been investigated in various studies, e.g., (Kaszkiel and Zobel, 2001; Callan, 1994; Hearst and Plaunt, 1993; Kaszkiel and Zobel, 1997). Besides the argument of decreasing the search space for subsequent answer extraction modules in QA, passage retrieval also improves standard IR techniques by "normalizing" textual units in terms of size which is especially important in cases where documents come from very diverse sources. IR is based on similarity measures between documents and queries and standard approaches have shortcomings when applying them to documents of various sizes and text types. Often there is a bias for certain types raising problems of discrimination between documents of different lengths and content densities. Passages on the other hand provide convenient units to be returned to the user avoiding such ranking difficulties (Kaszkiel and Zobel, 2001). For IR, passage-level evidence may be incorporated into document retrieval (Callan, 1994; Hearst and Plaunt, 1993) or passages may be used directly as retrieval unit (Kaszkiel and Zobel, 2001; Kaszkiel and Zobel, 1997). For QA only the latter is interesting and will be applied in our experiments.

Passages can be defined in various ways. The most obvious way is to use existing markup (explicit discourse information) to divide documents into smaller units. Unfortunately, such markup is not always available or ambiguous with other types of separators. For example, headers, list elements or table cells might be separated in the same way (for example using an empty line) as discourse related paragraphs. Also, the division into paragraphs may differ a lot depending on the source of the document. For example, Wikipedia entries are divided on various levels into rather small units whereas newspaper articles often include very long paragraphs.

There are several ways of automatically dividing documents into passages without relying on existing markup. One way is to search for linguistic clues that indicate a separation of consecutive text blocks. These clues may include lexical patterns and relations. We refer to such approaches as *semantically motivated* document segmentation. Another approach is to cut documents into arbitrary pieces ignoring any other type of information. For example, we can use fixed-sized windows to divide documents into passages of similar size. Such windows can be defined in terms of words and characters (Kaszkiel and Zobel, 2001; Monz, 2003) or sentences and paragraphs (Zobel et al., 1995; Llopis et al., 2002). It is also possible to allow varying window sizes and overlapping sections to be indexed (Kaszkiel and Zobel, 2001; Monz, 2003). In this case it is up to the IR engine

to decide which of the competing window types is preferred and it may even return overlapping sections multiple times.

In the following sections we will discuss two techniques of semantically motivated document segmentation and compare them to simple window-based techniques in terms of passage retrieval and QA performance.

2 Passage Retrieval in our QA system

Our QA system is an open-domain question answering system for Dutch. It includes two strategies: (1) A table-lookup strategy using fact databases that have been created off-line, and, (2) an "on-line" answer extraction strategy with passage retrieval and subsequent answer identification and ranking modules. We will only look at the second strategy as we are interested in the passage retrieval component and its impact on QA performance.

The passage retrieval component is implemented as an interface to several open-source IR engines. The query is generated from the given natural language question after question analysis. Keywords are sent to the IR engine(s) and results (in form of sentence IDs) are returned to the QA system.

In the experiments described here, we apply Zettair (Lester et al., 2006), an open-source IR engine developed by the search engine group at the RMIT University in Melbourne, Australia. It implements a very efficient standard IR engine with high retrieval performance according to our experiments with various alternative systems. Zettair is optimized for speed and is very efficient in both indexing and retrieval. The outstanding speed in indexing is very fortunate for our experiments in which we had to create various indexes with different document segmentation strategies.

3 Document Segmentation

We now discuss the different methods for document segmentation, starting with the semantically motivated ones and then looking at the window-based techniques.

3.1 Using Coreference Chains

Coreference is the relation which holds between two NPs both of which are interpreted as referring to the same unique referent in the context in which they occur ((Van Deemter and Kibble, 2000)). Since the coreference relation is an equivalence relation and consequently a transitive relation chains of coreferring entities can be detected in arbitrary documents. We can use these coreference chains to demarcate passages in the text. The assumption in this approach is that coreference chains mark semantically coherent passages, which are good candidates for splitting up documents.

Figure 1 illustrates chains detected by a resolution system in five successive sentences.

1. [Jim McClements en Susan Sandvig-Shobe]$_i$ hebben een onrechtmatig argument gebruikt.
2. [De Nederlandse scheidsrechter]$_j$ [Jacques de Koning]$_j$ bevestigt dit.
3. [Kuipers]$_k$ versloeg zondag in een rechtstreeks duel [Shani Davis]$_m$.
4. Toch werd [hij]$_k$ in de rangschikking achter [de Amerikaan]$_m$ geklasseerd.
5. [De twee hoofdarbiters]$_i$ verklaarden dat [Kuipers']$_k$ voorste schaats niet op de grond stond.

Cluster i (1,5): [Jim McClements en Susan Sandvig-Shobe] [De twee hoofdarbiters]

Cluster j (2): [De Nederlandse scheidsrechter] [Jacques de Koning]

Cluster k (3-5): [Kuipers] [hij] [Kuipers']

Cluster m (3,4): [Shani Davis] [de Amerikaan]

Figure 1: Example of coreference chains used for document segmentation

The coreferential units can then be used to form passages consisting of all sentences the coreference chain spans over, i.e. the boundaries of passages are sentences containing the first occurrence of the referent and the last occurrence of a referent. Thus, in the example in figure 1 we obtain four passages: 1) sentence one to sentence five, 2) sentence two, 3) sentence three to five, and, 4) sentence three and four. Note that such passages can be included in others and may overlap with yet others. Furthermore, there might be sentences which are not included in any chain which have to be handled by some other techniques.

For our purposes we used our own coreference resolution system which is based on information derived from Alpino, a wide-coverage dependency parser for Dutch (van Noord, 2006). We approached the task of coreference resolution as a

clustering-based ranking task. Some NP pairs are more likely to be coreferent than others. The system ranks possible antecedents for each anaphor considering syntactic features, semantic features and surface structure features from the anaphor and the candidate itself, as well as features from the cluster to which the candidate belongs. It picks the most likely candidate as the coreferring antecedent.

References relations are detected between pronouns, common nouns and named entities. The resolution system yields a precision of 67.9% and a recall of 45.6% (F-score = 54.5%) using MUC scores (Vilain et al., 1993) on the annotated test corpus developed by (Hoste, 2005) which consist of articles taken from KNACK, a Flemish weekly news magazine.

3.2 TextTiling

TextTiling is a well-known algorithm for segmenting texts into subtopic passages (Hearst, 1997). It is based on the assumption that a significant portion of a set of lexical items in use during the course of a given subtopic discussion changes when that subtopic in the text changes.

Topic shifts are found by searching for lexical co-occurrence patterns and comparing adjacent blocks. First the text is subdivided into pseudo-sentences of a predefined size rather than using syntactically-determined sentences. These pseudo-sentences are called token-sequences by Hearst.

The algorithm identifies discourse boundaries by calculating a score for each token-sequence gap. This score is based on two methods, block comparison and vocabulary introduction. The block comparison method compares adjacent blocks of text to see how similar they are according to how many words the adjacent blocks have in common. The vocabulary introduction method is based on how many new words were seen in the interval in which the token-sequence gap is the midpoint.

The boundaries are assumed to occur at the largest valleys in the graph that results from plotting the token-sequences against their scores. In this way the algorithm produces a flat subtopic structure from a given document.

3.3 Window-based

The simplest way of dividing documents into passages is to use a fixed-sized window. Here we do not take any discourse information nor semantic clue into account but split documents at arbitrary positions. Windows can be defined in various ways, in terms of characters, words or sentences. In our case it is important to keep sentences together because of the answer extraction component in our QA system that works on that level and expects complete sentences. Window-based segmentation techniques may be applied with various amounts of overlaps. The simplest method is to split documents into passages in a greedy way, starting a new passage immediately after the previous one (and starting the entire process at the beginning of each document)[1]. Another method is to allow some overlap between consecutive passages, i.e. starting a new passage at some position within the previous one. If we use the maximum possible overlap such an approach is usually called a "sliding window" in which the difference between two consecutive passages is only two basic units (sentences) - the first and the last one.

4 Experiments

4.1 Setup

For our experiments we applied the Dutch newspaper corpus used at the QA track at CLEF, the cross-language evaluation forum. It contains about 190,000 documents consisting of about 4,000,000 sentences (roughly 80 million words). As mentioned earlier, we applied the open-source IR engine, Zettair, in our experiments and used a language modeling metric with Dirichlet smoothing, which is implemented in the system.

The evaluation is based on 778 Dutch CLEF questions from the QA tracks in the years 2003 – 2005 which are annotated with their answers. We use simple matching of possible answer strings to determine if a passage is relevant for finding an accepted answer or not. Similarly, answer string matching is applied to evaluate the output of the entire QA system; i.e. an answer by the system is counted as correct if it is identical to one of the accepted answer strings without looking at the supporting sentence/passage. For evaluation we used the standard measure of MRR which is defined as follows:

[1] Note that in our approach we still keep the document boundaries intact, i.e. the segmentation ends at the end of each document and starts from scratch at the beginning of the next document. In this way, the last passage in a document may be smaller than the pre-defined fixed size.

$$MRR_{QA} = \frac{1}{N}\sum_{1}^{N}\frac{1}{rank(\text{first\_correct\_answer})}$$

Using the string matching strategy for evaluation this corresponds to the *lenient* MRR measures frequently used in the literature. *Strict* MRR scores (requiring a match with supporting documents) is less appropriate for our data coming from the CLEF QA tracks. In CLEF there are usually only a few participants and, therefore, only a small fraction of relevant documents are known for the given questions.

4.2 Evaluation of Passage Retrieval

There are various metrics that can be employed for evaluating passage retrieval. Commonly it is argued that passage retrieval for QA is merely a filtering task and ranking (precision) is less important than recall. Therefore, the measure of *redundancy* has been introduced which is defined as the average number of relevant passages retrieved per question (independent of any ranking). Passage retrieval is, of course, a bottleneck in QA systems that make use of such a component. The system has no chance to find an answer if the retrieval engine fails to return relevant passages. Therefore, another measure, *coverage* is often used in combination with redundancy. It is defined as the proportion of questions for which at least one relevant passage is found. In order to validate the use of these measures in our setup we experimented with retrieving various amounts of paragraphs. Figure 2 illustrates the relation of coverage and redundancy scores compared to the overall QA performance measured in terms of MRR scores.

From the figure we can conclude that coverage is more important than redundancy in our system. In other words, our QA system is quite good in finding appropriate answers if there is at least one relevant passage in the set of retrieved ones. Redundancy on the other hand does not seem to provide valuable insides for the end-to-end performance of our QA system.

However, our system also uses the passage retrieval score (and, hence, the ranking) as a clue for answer extraction. Therefore, other standard IR measures might be interesting for our investigations as well. The following three metrics are common in the IR literature.

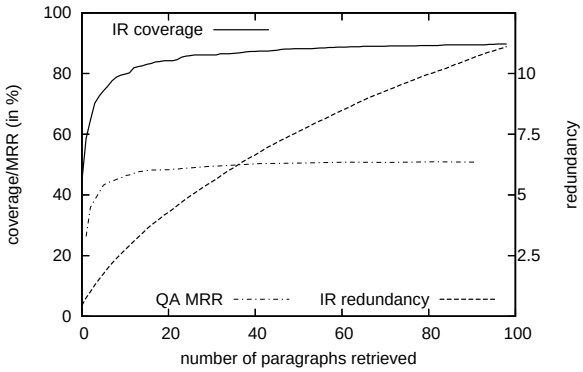

Figure 2: The correlation between coverage and redundancy and MRR_{QA} with varying numbers of paragraphs retrieved. Note that redundancy and coverage use different scales on the y-axis which makes them not directly comparable. The intention of this plot is to illustrate the tendency of both measures in comparison with QA performance.

Mean average precision (MAP): Average of precision scores for top k documents; MAP is the mean of these averages over all the N queries.

$$MAP = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{K}\sum_{k=1}^{K}P_n(1..k)$$

($P_n(1..k)$ is the precision of the top k documents retrieved for query q_n)

Uninterpolated average precision (UAP): Average of precision scores at each *relevant* document retrieved; UAP is the mean of these averages over the N queries.

$$UAP = \frac{1}{N}\sum_{n=1}^{N}\frac{1}{|D_r^n|}\sum_{k:d_k\in D_r^n}P_n(1..k)$$

(D_r^n is the set of relevant documents among the ones retrieved for question n)

Mean reciprocal ranks: The mean of the reciprocal rank of the first relevant passage retrieved.

$$MRR_{IR} = \frac{1}{N}\sum_{1}^{N}\frac{1}{rank(\text{first\_relevant\_passage})}$$

In figure 3 the correlation of these measures with the overall QA performance is illustrated.

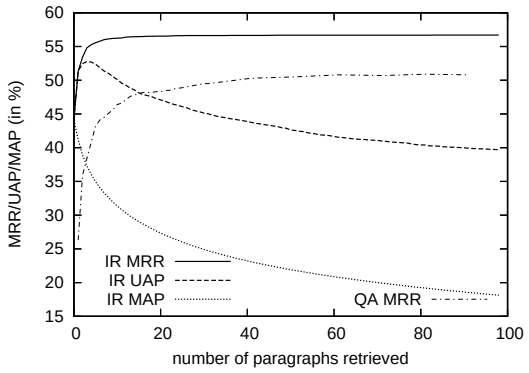

Figure 3: The correlation between IR evaluation measures (MAP, UAP and MRR_{IR}) and QA evaluation scores (MRR_{QA}) with varying numbers of paragraphs retrieved.

From the picture we can clearly see that the MRR_{IR} scores correlate the most with the QA evaluation scores when retrieving different numbers of paragraphs. This, again, confirms the importance of coverage as the MRR_{IR} score only takes the first relevant passage into account and ignores the fact that there might be more answers to be found in lower ranked passages. Hence, MRR_{IR} seems to be a good measure that combines coverage with an evaluation of the ranking and, therefore, we will use it as our main IR evaluation metric instead of coverage, redundancy, MAP & UAP.

4.3 Baselines

The CLEF newspaper corpus comes with paragraph markup which can easily be used as the segmentation granularity for passage retrieval. Table 1 shows the scores obtained by different baseline retrieval approaches using either sentences, paragraphs or documents as base units.

We can see from the results that document retrieval (used for QA) is clearly outperformed by both sentence and paragraph retrieval. Surprisingly, sentence retrieval works even better than paragraph retrieval when looking at the QA performance even though all IR evaluation measures (cov, red, MRR_{IR}) suggest a lower score. Note that MRR_{IR} is almost as good as MRR_{QA} for sentence retrieval whereas the difference between them is quite large for the other settings. This indicates the importance of narrowing down the search space for the answer extraction modules. The

				MRR		
	#sent	cov	red	IR	QA	CLEF
sent	16,737	0.784	2.95	0.490	**0.487**	**0.430**
par	80,046	0.842	4.17	0.565	0.483	0.416
doc	618,865	0.877	6.13	0.666	0.457	0.387

Table 1: Baselines with sentence (sent), paragraph (par) and document (doc) retrieval (20 units). MRR_{QA} is measured on the top 5 answers retrieved. *CLEF* is the accuracy of the QA system measured on the top answer provided by the system. *cov* refers to coverage and *red* refers to redundancy. *#sent* gives the total number of sentences included in the retrieved text units to give an impression about the amount of text to be processed by subsequent answer extraction modules.

amount of data to be processed is much smaller for sentence retrieval than for the other two while coverage is still reasonably high. The CLEF scores (accuracy measured on the top answer provided by the system) follow the same pattern. Here, the difference between sentence retrieval and document retrieval is even more apparent.

Certainly, the success of the retrieval component depends on the metric used for ranking documents as implemented in the IR engine. In order to verify the importance of document segmentation in a QA setting we also ran experiments with another standard metric implemented in Zettair, the Okapi BM-25 metric (Robertson et al., 1992). Similar to the previous setting using the LM metric, QA with paragraph retrieval (now yielding $MRR_{QA} = 0.460$) outperforms QA with document retrieval ($MRR_{QA} = 0.449$). However, sentence retrieval does not perform as well ($MRR_{QA} = 0.420$) which suggests that the Okapi metric is not suited for very small retrieval units. Still, the success of paragraph retrieval supports the advantage of passage retrieval compared to document retrieval and suggests potential QA performance gains with improved document segmentation strategies. In the remaining we only report results using the LM metric for retrieval due to its superior performance.

4.4 Semantically Motivated Passages

As described earlier, coreference chains can be used to extract semantically coherent passages from textual documents. In our experiments we used several settings for the integration of such passages in the retrieval engine. First of all, coref-

erence chains have been used as the only way of forming passages. Sentences which are not included in any passage are included as single-sentence passages. This settings is referred to as *sent/coref*.

In the second setting we restrict the passages in length. Coreference chains can be arbitrary long and, as we can see in the results in table 2, the IR engine tends to prefer long passages which is not desirable in the QA setting. Hence, we define the constraint that passages have to be longer than 200 characters and shorter than 1000. This setup is referred to as *sent/coref (200-1000)*.

In the third setting we combine paragraphs (using existing markup) and coreference chain passages including the length restriction. This is mainly to get rid of the single-sentence passages included in the previous settings. Note that all paragraphs are used even if all sentences within them are included in coreferential passages. Note also that in all settings passages may refer to overlapping text units as coreference chains may stretch over various overlapping passages of a document.

We did not perform an exhaustive optimization of the length restriction. However, we experimented with various settings and 200-1000 was the best performing one in our experiments. For illustration we include one additional experiment using a slightly different length constraint (200-400) in table 2.

For the document segmentation strategy using TextTiling we used a freely available implementation of that algorithm (the Perl Module `Lingua::EN::Segmenter::TextTiling` available at CPAN). Note that we do not include other passages in this approach (paragraphs using existing markup nor single-sentence passages).

Table 2 summarizes the scores obtained by the various settings when applied for passage retrieval and when embedded into the QA system.

It is worth noting that including coreferential chains without length restriction forced the retrieval engine to return a lot of very long passages which resulted in a degraded QA performance (also in terms of processing time which is not shown here). The combination of paragraphs and coreferential passages with length restrictions produced MRR_{QA} scores above the baseline. However, these improvements are not statistically significant according to the Wilcoxon matched-pair signed-ranks test and looking at the corresponding CLEF scores we can even see a slight drop in performance. Applying TextTiling yielded improved scores in both passage retrieval and QA performance (MRR_{QA} and CLEF). The MRR_{QA} improvement is statistically significant according to the same test.

		MRR		
	#sent	IR	QA	CLEF
sent/coref	490,968	**0.604**	0.469	0.405
sent/coref (200-1000)	76,865	0.535	0.462	0.395
par+coref (200-1000)	82,378	0.560	**0.493**	0.426
par+coref (200-400)	67,580	0.555	**0.489**	0.422
TextTiling	107,879	**0.586**	△ **0.503**	**0.434**

Table 2: Passage retrieval with document segmentation using coreference chains and TextTiling (retrieving a maximum of 20 passages; △ means significant with $p < 0.05$ and Wilcoxon Matched-pair Signed-Ranks Test compared to paragraph baseline – only tested for MRR_{QA})

4.5 Window-based Passages

In comparison to the semantically motivated passages discussed above we also looked at simple window-based passages as described earlier. Here we do not consider any linguistic clues for dividing the documents besides the sentence and document boundaries. Table 3 summarizes the results obtained for various fixed-sized windows used for document segmentation.

		MRR		
	#sent	IR	QA	CLEF
2 sentences	33468	0.545	△ **0.506**	**0.443**
3 sentences	50190	0.554	**0.504**	**0.436**
4 sentences	66800	**0.581**	△ **0.512**	**0.447**
5 sentences	83575	**0.588**	**0.493**	0.422
6 sentences	100110	**0.583**	**0.489**	0.423
7 sentences	116872	**0.572**	**0.491**	0.422
8 sentences	133504	**0.577**	0.481	0.409
9 sentences	150156	**0.578**	0.475	0.405
10 sentences	166810	**0.596**	0.470	0.396

Table 3: Passage retrieval with window-based document segmentation (△ means significant with $p < 0.05$ and Wilcoxon Matched-pair Signed-Ranks Test)

Surprisingly, we can see that window-based segmentation approaches with small sizes between 2 and 7 sentences yield improved scores compared to the baseline. Two of the improvements (using 2-sentence passages and 4-sentence passages) are statistically significant. Three settings also out-

perform the best semantically motivated segmentation approach. This result was unexpected especially considering the naive way of splitting documents into parts disregarding any discourse structure (besides document boundaries) and other semantic clues.

We did another experiment using window-based segmentation and a sliding window approach. Here, fixed-sized passages are included starting at every point in the document and, hence, various overlapping passages are included in the index. In this way we split documents at various points and leave it to the IR engine to select the most appropriate ones for a given query. The results are shown in table 4.

		MRR		
	#sent	IR	QA	CLEF
2 sent (sliding)	29095	0.548	△ **0.516**	**0.456**
3 sent (sliding)	36415	0.549	**0.484**	0.411
4 sent (sliding)	41565	0.546	0.476	0.409
5 sent (sliding)	45737	0.534	0.465	0.403
6 sent (sliding)	49091	0.528	0.454	0.390
7 sent (sliding)	51823	0.529	0.439	0.372
8 sent (sliding)	54600	0.535	0.428	0.360
9 sent (sliding)	57071	0.531	0.420	0.351
10 sent (sliding)	59352	0.542	0.420	0.354

Table 4: Passage retrieval with window-based document segmentation and a sliding window

Again, we see a significant improvement with 2-sentence passages (the overall best score so far) but the performance degrades when increasing the window size. Note that the number of sentences retrieved is growing very slowly for larger windows. This is because more and more of the overlapping regions are retrieved and, hence, the total number of unique sentences does not grow with the size of the window as we have seen in the non-sliding approach.

5 Discussion & Conclusions

Our experiments show that passage retrieval is indeed different to general document retrieval. Improved retrieval scores do not necessarily lead to better QA performance. Important for QA is to reduce the search space for subsequent answer extraction modules and, hence, passage retrieval has to balance retrieval accuracy and retrieval size. In our setup it seems to be preferable to return very small units with a reasonable coverage. For this, index-time passaging is very effective.

In this study we were especially interested in semantically motivated approaches to document segmentation. In particular, two techniques have been investigated, one using the well-known TextTiling algorithm and one using coreference chains for passage boundary detection. We compared them to simple window-based techniques using various sizes. From our experiments we can conclude that simple document segmentation techniques using small fixed-sized windows work best among the ones tested here. Semantically motivated passages in the retrieval component helped to slightly improve QA performance but do not justify the effort spent in producing them. One of the main reasons for the failure of using coreference chains for segmentation might be the fact that this approach produces many overlapping passages which does not seem to be favorable for passage retrieval. This can also be seen in the sliding window approach which did not perform as well as the one without overlapping units (except for two-sentence passages). In conclusion, *uniformity* in terms of length and *uniqueness* (in terms of non-overlapping contents) seem to be more important than semantic coherence for one-step passage retrieval in QA. A future direction could be to test an approach that balances both a uniform document segmentation and semantic coherence.

References

Callan, James P. 1994. Passage-level evidence in document retrieval. In *SIGIR '94: Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 302–310, New York, NY, USA. Springer-Verlag New York, Inc.

Greenwood, Mark A. 2004. Using pertainyms to improve passage retrieval for questions requesting information about a location. In *Proceedings of the Workshop on Information Retrieval for Question Answering (SIGIR 2004)*, Sheffield, UK.

Hearst, Marti A. and Christian Plaunt. 1993. Subtopic structuring for full-length document access. In *Research and Development in Information Retrieval*, pages 59–68.

Hearst, Marti A. 1997. Texttiling: Segmenting text into multi-paragraph subtopic passages. *Computational Linguistics*, 23(1):33–64.

Hoste, V. 2005. *Optimization Issues in Machine Learning of Coreference Resolution*. Ph.D. thesis, University of Antwerp.

Kaszkiel, Marcin and Justin Zobel. 1997. Passage retrieval revisited. In *SIGIR '97: Proceedings of the 20th annual international ACM SIGIR conference on*

Research and development in information retrieval, pages 178–185, New York, NY, USA. ACM Press.

Kaszkiel, Marcin and Justin Zobel. 2001. Effective ranking with arbitrary passages. *Journal of the American Society of Information Science*, 52(4):344–364.

Lester, Nicholas, Hugh Williams, Justin Zobel, Falk Scholer, Dirk Bahle, John Yiannis, Bodo von Billerbeck, Steven Garcia, and William Webber. 2006. The Zettair search engine. http://www.seg.rmit.edu.au/zettair/.

Llopis, F., J. Vicedo, and A. Ferrández. 2002. Passage selection to improve question answering. In *Proceedings of the COLING 2002 Workshop on Multilingual Summarization and Question Answering*.

Moldovan, D., S. Harabagiu, M. Pasca, R. Mihalcea, R. Girju, R. Goodrum, and V. Rus. 2000. The structure and performance of an open-domain question answering system.

Monz, Christof. 2003. *From Document Retrieval to Question Answering*. Ph.D. thesis, University of Amsterdam.

Roberts, Ian and Robert Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *Proceedings of 26th European Conference on Information Retrieval*.

Robertson, Stephen E., Steve Walker, Micheline Hancock-Beaulieu, Aarron Gull, and Marianna Lau. 1992. Okapi at TREC-3. In *Text REtrieval Conference*, pages 21–30.

Tellex, S., B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *Proceedings of the SIGIR conference on Research and development in informaion retrieval*, pages 41–47. ACM Press.

Van Deemter, K. and R. Kibble. 2000. On coreferring: Coreference in muc and related annotation schemes. *Computational Linguistics*, 26(4):629–637.

van Noord, Gertjan. 2006. **A**t **L**ast **P**arsing **I**s **N**ow **O**perational. In *TALN 2006 Verbum Ex Machina, Actes De La 13e Conference sur Le Traitement Automatique des Langues naturelles*, pages 20–42, Leuven.

Vilain, M., J. Burger, J. Aberdeen, D. Connolly, and L. Hirschman. 1993. A model-theoretic coreference scoring scheme. In *Proceedings of the 6th conference on Message understanding (MUC 6)*, pages 45–52.

Zobel, Justin, Alistair Moffat, Ross Wilkinson, and Ron Sacks-Davis. 1995. Efficient retrieval of partial documents. *Information Processing and Management*, 31(3):361–377.

Passage Retrieval for Question Answering using Sliding Windows

Mahboob Alam Khalid
ISLA, University of Amsterdam
mahboob@science.uva.nl

Suzan Verberne
Radboud University Nijmegen
s.verberne@let.ru.nl

Abstract

The information retrieval (IR) community has investigated many different techniques to retrieve passages from large collections of documents for question answering (QA). In this paper, we specifically examine and quantitatively compare the impact of passage retrieval for QA using sliding windows and disjoint windows. We consider two different data sets, the TREC 2002–2003 QA data set, and 93 *why*-questions against INEX Wikipedia. We discovered that, compared to disjoint windows, using sliding windows results in improved performance of TREC-QA in terms of TDRR, and in improved performance of *why*-QA in terms of success@n and MRR.

1 Introduction

In question answering (QA), text passages are an important intermediary between full documents and exact answers. They form a very natural unit of response for QA systems (Tellex et al., 2003) and it is known from user studies that users prefer answers to be embedded in paragraph-sized chunks (Lin et al., 2003) because they can provide the context of an answer. Therefore, almost all state-of-the-art QA systems implement some technique for extracting paragraph-sized fragments of text from a large corpus.

Most QA systems have a pipeline architecture consisting of at least three components: question analysis, document/passage retrieval, and answer extraction (Hirschman and Gaizauskas, 2001; Voorhees, 2001). The quality of a QA system heavily depends on the effectiveness of the integrated retrieval system (second step of the pipeline): if a retrieval system fails to find any relevant documents for a question, further processing steps to extract an answer will inevitably fail too (Monz, 2003). This motivates the need to study passage retrieval for QA.

There are two common approaches to retrieving passages from a corpus: one is to index each passage as separate document and retrieve them as such. The other option is to first retrieve relevant documents for a given question and then retrieve passages from the retrieved documents. The passages themselves can vary in size and degree of overlap. Their size can be fixed as a number of words or characters, or varying with the semantic content (Hearst and Plaunt, 1993) or the structure of the text (Callan, 1994). The overlap between two adjacent passages can be either zero, in which case we speak of *disjoint passages*, or the passages may be overlapping, which we refer to as *sliding passages*.

In this paper, we compare the effectiveness of several passage retrieval techniques with respect to their usefulness for QA. Our main interest is the contribution of sliding passages as apposed to disjoint passages, and we will experiment with a number of retrieval models. We evaluate the retrieval approaches on two different QA tasks: (1) factoid-QA, as defined by the test collection provided by TREC (Voorhees, 2002; Voorhees, 2003), and (2) a relatively new problem in the QA field: that of answering *why*-questions (*why*-QA).

The remainder of the paper is organized as follows. In the next section, we describe related work on passage retrieval for QA and we motivate what the main contribution of the current paper is. In

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 26–33
Manchester, UK. August 2008

section 3 we describe our general set-up for passage retrieval in both QA tasks that we consider. In section 4, we present the results of the experiments on TREC-QA data, and in section 5 we present our results on *why*-QA. Section 6 gives an overall conclusion.

2 Related work

The use of passage retrieval for QA has been studied before. For example, (Tellex et al., 2003) performed a quantitative evaluation of passage retrieval algorithms for QA. They compared different passage retrieval algorithms in the context of their QA system. Their system first returns a ranked list of 200 documents and then applies different passage retrieval algorithms to the retrieved documents. They find that the performance of passage retrieval depends on the performance of the pre-applied document retrieval step, and therefore they suggest that document and passage retrieval technology should be developed independently.

A similar message is conveyed by (Roberts and Gaizauskas, 2004). They investigate different approaches to passage retrieval for QA. They identify each paragraph as a seperate passage. They find that the optimal approach is to allow multiple passages per document to be returned and to score passages independently of their source document.

(Tiedemann, 2007) studies the impact of document segmentation approaches on the retrieval performance of IR for Dutch QA. He finds that segmentation based on document structure such as the use of paragraph markup (discourse-based segmentation) works well with standard information retrieval techniques. He tests various other techniques for document segmentation and various passage sizes. In his experimental setting, larger text units (such as documents) produce better performance in passage retrieval. Tiedemann compares different sizes of discourse-based segmentation: sentences, paragraphs and documents. He finds that larger text units result in a large search space for subsequent QA modules and hence reduce the overall performance of the QA system. That is why we do not conduct experiments with different passage sizes in this paper: it is difficult to measure the outcome of such experiments independently of the specific answer extraction system. We adopt Tiedemann's best strategy of document segmentation strategy, i.e., paragraph-based, but with equally sized passages instead.

3 General experiment set-up

The main purpose of our experiments is to study the contribution of sliding windows as apposed to disjoint windows in the context of QA. Therefore, in our experiment setup, we have kept fixed the other segmentation variables, passage size and degree of overlap. We set out to examine two different strategies of document segmentation (disjoint and sliding passages) with a number of retrieval models for two different QA tasks: TREC factoid-QA and *why*-QA.

3.1 Retrieval models

We use the Lemur retrieval engine[1] for passage retrieval because it provides a flexible support for different types of retrieval models including vector space models and language models. In this paper we have selected two vector space models: TFIDF and Okapi BM25 (Robertson and Walker, 1999), and one language model based on Kullback-Leibler (KL) divergence (Lafferty and Zhai, 2001).

The TFIDF weighting scheme is often used in information retrieval. There are several variations of the TFIDF weighting scheme that can effect the performance significantly. The Lemur toolkit provides a variant of the TFIDF model based on the Okapi TF formula (Robertson et al., 1995).

Lemur also provides the implementation of the original Okapi BM25 model, and we have used this model with default values of 1.2 for $k1$, 0.75 for b and 7 for $k3$ as suggested by (Robertson and Walker, 1999). The KL-divergence retrieval model, which implements the cross entropy of the query model with respect to the document model, is a standard metric for comparing distributions, which has proven to work well in IR experiments in the past. To address the data sparseness problem during model estimation, we use the Dirichlet smoothing method (Zhai and Lafferty, 2004) with default parameter values provided in the Lemur toolkit.

Currently, however, the Lemur [2] does not support direct passage retrieval. For these experiments, therefore, we first need to segment documents into passages before indexing them into the

[1] Lemur toolkit: http://www.lemurproject.org

[2] Lemur and Indri are different search engines. Indri provides the $\#passage$ operator, but it doesn't consider paragraph boundaries or sentence boundaries for constructing passages.

Lemur retrieval engine. Our segmenting strategy is explained below.

3.2 Passage identification

For our experiments, we take into account two different corpora: AQUAINT and the Wikipedia XML corpus as used in INEX (Denoyer and Gallinari, 2006). The AQUAINT corpus consists of news articles from the Associated Press, New York Times, and Xinhua News Agency (English version) from 1996 to 2000. The Wikipedia XML collection consists of 659,388 articles as they occured in the online Wikipedia in the summer of 2006. As we have discussed in Section 2, (Tiedemann, 2007) discovered that discourse-based segmentation into paragraphs works well with standard information retrieval techniques. They also observe that larger retrieval units produce better results for passage retrieval, since larger units have higher chance to cover the required information. Therefore, we decide to segment each document into similar sized passages while taking into account complete paragraphs only.

For document segmentation, our method first detects sentences in the text using punctuation marks as separators, and then paragraphs using empty lines as separators. Sentence boundaries are necessary because we aim at retrieving passages that do not contain any broken sentences. The required passages are identified by aligning over paragraph boundaries (merging paragraphs into units until they have the required length ,i.e. 500 characters). The disjoint passages do not share any content with each other, and the sliding passages slide with the difference of one paragraph boundary, i.e., we start forming a new passage from beginning of each paragraph of the document. If paragraph boundaries are not detected, then these sliding passages are half-overlapped with each other.

For the Wikipedia XML corpus, we have found that documents have already been annotated with <p> elements. Thus we consider these elemens as paragraph boundaries instead of empty lines as we did for the AQAINT corpus. We observe that some textual parts of the documents are not covered by the XML paragraph boundaries. Therefore we have extended the existing paragraph boundaries such that the missing text fragments become part of the paragraphs.

We split both corpora into disjoint and sliding windows as we have discussed above. After splitting the 1.03M documents of the AQUAINT-1 collection we have 14.2M sliding passages, and 4.82M disjoint passages. And similarly we got 4.1M sliding passages and 2M disjoint passages from the Wikipedia XML collection of 659,388 documents.

3.3 Evaluation metrics

For our experiments, we use the following metrics for evaluation:

Mean reciprocal rank (MRR) at n is the mean (calculated over all questions) of the reciprocal rank (which is 1 divided by the rank ordinal) of the highest ranked relevant (i.e. answer bearing) passage. RR is zero for a question if no relevant passage is returned by the system at limit n.

Success at n for a question is 1 if the answer to this question is found in top n passages fetched up by our system. Success@n is averaged over all questions.

Total document reciprocal rank (TDRR) (Bilotti et al., 2004) is the sum of all reciprocal ranks of all answer bearing passages per question (averaged over all questions). The value of TDRR is maximum if all retrieved passages are relevant. TDRR is an extension of MRR that favors a system that ranks more that one relevant passage higher than all non-relevant passages. This way, TDRR extends MRR with a notion of recall.

When we compare retrieval performance of two retrieval settings (such as the use of *disjoint* versus *sliding* windows), then we obtain a list of paired scores. That's why we use the Wilcoxon signed-rank test to show the statistical significance of the improvements.

In summary, we experiment with three retrieval models in Lemur: TFIDF, Okapi, and a language model based on the Kullback-Leibler divergence. For each of these retrieval models, we evaluate the use of both sliding and disjoint passages. This makes a total of six retrieval settings.

4 Evaluating passage retrieval for TREC-QA

As test collection for factoid QA, we use a standard set of 822 question/answer pairs from the TREC

QA tasks of 2002-2003. For evaluation of the passage retrieval approaches that we consider, we compute strict scores as defined by (Tellex et al., 2003). Strict scoring means that a retrieved passage is considered relevant if the passage not only matches one of the answer patterns provided by NIST, but its associated document is also listed as one of the relevant documents assessed by NIST. (Bilotti et al., 2004) have reviewed 109 factoid questions of the TREC-2002 task and they have extended the existing set of relevant documents by adding more relevant documents. We have also included this extended list of relevant documents for these questions in our experiment setup.

We evaluate the impact of disjoint and sliding windows on passage retrieval for QA using three different retrieval models, using the MRR@n, Success@n and TDRR@n metrics as described in section 3.3. Table 1 shows the evaluation results (best scores for each measure in bold face). The experiment results show that language model based on Kullback-Leibler divergence shows better performance than two vector space models for both types of windows retrieval according to MRR, success@n and TDRR evaluation metrics.

4.1 Discussion

In a pipeline QA system, the answer extraction module depends on the performance of passage retrieval. If more answer bearing passages are provided in the stream, then there is a high chance of selecting the correct answer from the stream in later stages of QA. (Roberts and Gaizauskas, 2004) have also discussed the importance of this aspect of passage retrieval for QA. They have measured the *answer redundancy* of a retrieval system which measures how many answer bearing passages are returned per question at limit n. (Tiedemann, 2007) have also used this metric and argue that high *redundancy* is desired to make it easier for the answer extraction module to spot possible answers. We consider TDRR as the most important measure for the passage retrieval task since it does not only measure the *redundancy* of a retrieval system but also measures how much improvement there is in returning the relevant passages at top ranks.

According to TDRR@n in table 1, retrieval of sliding windows outperforms retrieval of disjoint windows at all limits of n for all retrieval models. For $n = 100$, the improvement is significant at $p = 0.01$ level. This high value of TDRR@n suggests that segmenting the documents into sliding windows is a better choice in order to return as many relevant passages as possible at top ranks.

If we consider Success@n as evaluation measure instead of TDRR, retrieval of disjoint windows outperforms retrieval of sliding windows. We think that one of the reasons for this behaviour is that since sliding windows overlap with their neighbours, they are more pair-wise similar than disjoint windows. Therefore, it is possible that for some non-answered questions many irrelevant passages are returned at top ranks and that relevant passages are surpressed down.

5 Evaluating passage retrieval for *why*-QA

In the previous section, we showed that for TREC data, the choice of the retrieval model and the type of windows to be retrieved influence on the retrieval performance. We found that for the TREC data, a language modeling approach (based on Kullback-Leibler divergence) on sliding windows gives the best results in terms of TDRR. In this section, we aim to find out what the optimal passage retrieval approach is for a very different type of QA, namely *why*-QA.

5.1 Background of *why*-QA system development

In (Verberne et al., 2008), we present an approach for *why*-QA that is based on paragraph retrieval from the INEX Wikipedia corpus (Denoyer and Gallinari, 2006). Our system for *why*-QA consists of two modules: a passage retrieval module and a re-ranking module. In earlier retrieval experiments, we used the Wumpus retrieval system (Buttcher, 2007), and we defined passages simply by the XML paragraph markup <p>. Passage ranking in Wumpus is done by the QAP passage scoring algorithm (Buttcher et al., 2004).

The second module of our *why*-system is a re-ranking step that uses syntactic features of the question and the retrieved answers for adapting the scores of the answers and changing the ranking order. The weights of the re-ranking features have been optimized by training on our question answer data in five folds[3] using a genetic algorithm. We let Wumpus retrieve and rank 150 paragraphs per

[3] In five turns, we tune the feature weights on four of the five folds and evaluate them on the fifth

Table 1: Results for passage retrieval for TREC-QA using disjoint windows (DW) and sliding windows (SW). ** indicates a significant improvements of sliding windows over disjoint windows at the $p = 0.01$ level.

		MRR		Success@n		TDRR	
n	retrieval model	DW	SW	DW	SW	DW	SW
10	TFIDF	0.327	0.326	51.8%	50.1%	0.465	0.637
	Okapi	0.322	0.328	51.9%	51.2%	0.459	0.649
	KL	**0.355**	0.345	**55.7**%	51.3%	0.518	**0.710**
100	TFIDF	0.336	0.386	54.1%	53.3%	0.517	0.819**
	Okapi	0.333	0.339	**77.0**%	76.2%	0.535	0.835**
	KL	**0.363**	0.353	**77.1**%	75.2%	0.525	**0.902****

question. This number of 150 answers was chosen as a trade-off between covering as many as possible of the relevant answers retrieved by Wumpus, and the system load that was needed for automatic syntactic analysis of all answers in the second (re-ranking) module of the system. For evaluation of the results, we performed manual assessment of all answers retrieved, starting at the highest-ranked answer and ending as soon as we encountered a relevant answer[4].

The results for our original *why*-system are in Table 2. We show the results in terms of success@n and MRR@n. As opposed to the evaluation of TREC-QA, we do not consider TDRR as evaluation measure for experiments on *why*-QA. This is because in *why*-QA, we are only interested in the top-ranked answer-bearing passage. For calculating TDRR, assessment of all 150 retrieved answers would be necessary.

Table 2 shows that success@150 for the retrieval module (Wumpus/QAP) is 73.1%. This means that for 26.9% of the questions, no relevant answer is retrieved in the first module. Re-ranking the answers cannot increase MRR for these questions, since none of the 150 answers in the result list is relevant. We consider a success@150 score of 73.1% to be quite low. We aim to improve the performance of our system by optimizing its first module, passage retrieval.

We experiment with a number of passage retrieval approaches in order to reach better retrieval in the first module of our system. We aim to find out which type of retrieval model and what window type (disjoint or sliding) gives optimal results for retrieving passages relevant to *why*-questions. If the retrieval performance indeed goes up, we will apply our re-ranking module to the newly retrieved data to see what overall system performance we can reach with the new retrieval approach.

[4]We didn't need to assess the tail since we were only interested in the highest-ranked relevant answer for calculating MRR and success@n

5.2 Data and evaluation setup

For development and testing purposes, we use the Webclopedia question set by (Hovy et al., 2002). This set contains questions that were asked to the online QA system `answers.com`. 805 of these questions are *why*-questions. We manually inspect a sample of 400 of the Webclopedia *why*-questions. Of these, 93 have an answer in the Wikipedia XML corpus (see section 3). Manual extraction of one correct answer for each of these questions results in a set of 93 *why*-questions and their reference answer.

In order to be able to do fast evaluation of the different evaluation settings, we manually create an answer pattern for each of the questions in our set. These answer patterns are based on a set of 93 reference answers (one answer per question) that we have manually extracted from the Wikipedia corpus. An answer pattern is a regular expression that defines which of the retrieved passages are considered a relevant answer to the input question.

As opposed to the answer patterns provided by NIST for the evaluation of factoid QA (see section 4), our answer patterns for *why*-questions are relatively strict. A *why*-answer can be formulated in many different ways with different words, which may not all be in the answer pattern. For a factoid question such as "When was John Lennon born?", the answer is only one phrase, and the answer pattern is short and unambiguous, i.e. `/1940/`. However, if we consider the *why*-question "Why are some organ transplants unsuccessful?", the answer pattern cannot be stated in one phrase. For

Table 2: Results for the original *why*-QA pipeline system

	success@10	success@150	MRR@150
Wumpus/QAP Retrieval	43.0%	73.1%	0.260
+ Re-ranking module	54.8%	73.1%	0.380

this example, we created the following answer pattern based on the pre-extracted reference answer[5]: `/.*immune system.*foreign tissues.*destroy.*/`. It is however possible that a relevant answer is formulated in a way that does not match this regular expression. Thus, the use of answer patterns for the evaluation of *why*-QA leads to conservative results: some relevant answers may be missed in the evaluation procedure.

After applying the answer patterns, we count the questions that have at least one relevant answer in the top 10 and the top 150 of the results (success@10, success@150). For the highest ranked relevant answer per question, we determine the reciprocal rank (RR). If there is no correct answer retrieved by the system at $n = 150$, the RR is 0. Over all questions, we calculate the MRR@150.

5.3 Passage retrieval results

We segment and index the Wikipedia corpus as described in section 3 and run all six retrieval settings on our set of 93 *why*-questions. For consistent evaluation, we applied the answer patterns that we created to the newly retrieved Lemur data as well as to the original Wumpus output.

The retrieval results for all settings are in Table 3. We show both success@10 and success@150, and MRR@150 for each setting. Success@150 is important if we consider the current results as input for the re-ranking module. As explained before, re-ranking can only be successful if at least one relevant answer is retrieved by the retrieval module. For each measure (s@10, s@150 and MRR@150), the score of the highest-scoring setting is printed in bold face.

As expected, the evaluation of the Wumpus data with the use of answer patterns gives somewhat lower scores than evaluation based on manual assessment of all answers (table 2). This confirms our idea that the use of answer patterns for *why*-QA leads to conservative results. Thus we can state that the Lemur scores shown in table 3 are not overestimated and therefore reliable.

Since we are using the output of the passage retrieval module as input for our re-ranking module, we are mainly interested in the scores for success@150. For the four retrieval models, we see that TFIDF seems to score somewhat better on retrieving sliding windows in terms of success@150 than Okapi and the Kullback-Leibler language model. On the other hand, Kullback-Leibler and QAP seem to perform better on retrieving disjoint windows. However, these differences are not significant at the $p = 0.01$ level. For the differences between disjoint and sliding windows for all retrieval models together, we see that retrieval of sliding windows gives significantly better results than disjoint windows in terms of success@150 ($p < 0.001$).

5.4 The influence of passage retrieval on our pipeline system

As described in section 5.1, our system is a pipeline: after passage retrieval, we apply a re-ranking module that uses syntactic information for re-scoring the results from the retrieval module. As input for our re-ranking module we use the output of the retrieval setting with the highest success@150 score: Lemur/TFIDF on sliding windows. For 81.7% of the questions in our set, Lemur/TFIDF retrieved an answer in the top-150. This means that the maximum success@10 score that we can obtain by re-ranking is 81.7%.

For weighting the feature values, we re-use the weights that we had earlier found from training on our set of 93 questions and the 150 answers that were retrieved by Wumpus. We again take into account five-fold cross validation for evaluation. For a detailed description of our re-ranking module and the syntactic features that we exploit, we refer to (Verberne et al., 2008).

The results from re-ranking are in Table 4. In the table, four system versions are compared: (1) the original Wumpus/QAP module, (2) the original *why*-pipeline system: Wumpus/QAP with re-ranking, (3) TFIDF-sliding and (4) the new

[5]The pre-extracted reference answer is: "This is because a normal healthy human immune system can distinguish foreign tissues and attempts to destroy them, just as it attempts to destroy infective organisms such as bacteria and viruses."

Table 3: Results for passage retrieval on *why*-questions against Wikipedia using disjoint windows (DW) and sliding windows (SW)

	Success@10		Success@150		MRR@150	
Retrieval model	DW	SW	DW	SW	DW	SW
Baseline: Wumpus/QAP	40.9%		72.0%		0.229	
Lemur/TFIDF	43.0%	45.2%	71.1%	**81.7**%	0.247	**0.338**
Lemur/Okapi	41.9%	44.1%	67.7%	79.6%	0.243	0.320
Lemur/KL	48.9%	**50.0**%	72.8%	77.2%	0.263	0.324

pipeline system: TFIDF-sliding with re-ranking. We again show MRR, success@10 and success@150. For each measure, the score of the highest-scoring setting is printed in bold face.

After applying our re-ranking module (right bottom setting), we find a significant improvement over bare TFIDF (left bottom setting). In terms of MRR, we also see an improvement over the results that we had obtained by re-ranking the Wumpus/QAP output (right top setting). However, success@10 does not show significant improvement. The improvement that the re-ranking module gives is smaller for the TFIDF retrieval results (MRR goes from 0.338 to 0.359) than for the QAP results (MRR increases from 0.260 to 0.328). We suspect that this may be due to the fact that we used feature weights for re-ranking that we had earlier obtained from training on the Wumpus/QAP data (see section 5.4). It would be better to re-train our feature weights on the Lemur data. Probably, re-ranking can then make a bigger contribution than it does now for the Lemur data.

6 Overall conclusion

In this paper we have investigated the contribution of sliding windows as apposed to disjoint windows with different retrieval modules for two different QA tasks: the TREC-QA 2002–2003 task and *why*-QA.

For the TREC factoid-QA task, we have found that retrieval of sliding windows outperfoms retrieval of disjoint windows in returning as many relevant passages as possible on top ranks (according to the TDRR metric). The experimental results show that a language model based on Kullback-Leibler divergence gives better performance than two vector space models for both types of windows retrieval according to MRR, success@n and TDRR evaluation metrics. We found that the number of answered questions (success@n) was slightly lower when we used sliding windows for passage retrieval than disjoint windows, but we think one of the reasons is that sliding windows are more homogeneous than disjoint windows, and therefore for some questions more irrelevant passages are returned at top ranks and relevant passages are surpressed down.

For the task of retrieving answers to *why*-questions from Wikipedia data, we found that the best retrieval model is TFIDF, and sliding windows give significantly better results than disjoint windows. We also found better performance for our complete *why*-pipeline system after applying our existing re-ranking module to the passages retrieved with TFIDF-sliding.

In general, we find that for QA, sliding windows give better results than disjoint windows in the passage retrieval step. The best scoring retrieval model depends on the task under consideration, because the nature of the documents and question sets differ. This shows that for each specific QA task, different retrieval models should be considered.

In the future, we aim to boost passage retrieval for QA even more by applying query expansion techniques that are specific to the QA tasks that we consider, i.e. TREC factoid-QA and *why*-QA.

References

Bilotti, M.W., B. Katz, and J. Lin. 2004. What works better for question answering: Stemming or morphological query expansion. In *Proceedings of the SIGIR 2004 Workshop IR4QA: Information Retrieval for Question Answering, July*.

Buttcher, S., C.L.A. Clarke, and G.V. Cormack. 2004. Domain-specific synonym expansion and validation for biomedical information retrieval (multitext experiments for trec 2004).

Buttcher, S. 2007. The wumpus search engine. http://www.wumpus-search.org/.

Callan, James P. 1994. Passage-level evidence in document retrieval. In *SIGIR*, pages 302–310.

Table 4: Results for the *why*-QA pipeline system for best-scoring passage retrieval setting compared against the Wumpus baseline, for both bare retrieval and the complete system with re-ranking

	Success@10		Success@150		MRR	
Retrieval model	Bare	+Re-rank	Bare	+Re-rank	Bare	+Re-rank
Baseline: Wumpus/QAP-disjoint	43.0%	54.8%	73.1%	73.1%	0.260	0.328
Lemur/TFIDF-sliding	45.2%	**55.9**%	81.7%	81.7%	0.338	**0.359**

Denoyer, L. and P. Gallinari. 2006. The Wikipedia XML corpus. *ACM SIGIR Forum*, 40(1):64–69.

Hearst, Marti A. and Christian Plaunt. 1993. Subtopic structuring for full-length document access. In *ACM-SIGIR, 1993*, pages 59–68.

Hirschman, L. and R. Gaizauskas. 2001. Natural language question answering: the view from here. *Nat. Lang. Eng.*, pages 275–300.

Hovy, E.H., U. Hermjakob, and D. Ravichandran. 2002. A question/answer typology with surface text patterns. In *Proceedings of the Human Language Technology conference (HLT)*, San Diego, CA.

Lafferty, J. and C. Zhai. 2001. Document language models, query models, and risk minimization for information retrieval. In *In Proceedings of SIGIR'01*, pages 111–119.

Lin, J., D. Quan, V. Sinha, K. Bakshi, D. Huynh, B. Katz, and D.R. Karger. 2003. The role of context in question answering systems. *Conference on Human Factors in Computing Systems*, pages 1006–1007.

Monz, Christof. 2003. Document retrieval in the context of question answering. In *ECIR*, pages 571–579.

Roberts, I. and R. Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In *In Proceedings of ECIR , 2004.*

Robertson, Stephen E. and Steve Walker. 1999. Okapi/keenbow at trec-8. In *Text Retrieval Conference*.

Robertson, Stephen E., Steve Walker, Micheline Hancock-Beaulieu, and Gatford M. 1995. Okapi at trec-3. In *Text Retrieval Conference*, pages 109–26.

Tellex, S., B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *In SIGIR conference on Research and development in informaion retrieval, 2003*, pages 41–47.

Tiedemann, Jörg. 2007. Comparing document segmentation strategies for passage retrieval in question answering. *In Proceedings of RANLP 07, Borovets, Bulgaria*.

Verberne, Suzan, Lou Boves, Nelleke Oostdijk, and Peter-Arno Coppen. 2008. Using Syntactic Information for Improving Why-Question Answering. In *Proceedings of The 22nd International Conference on Computational Linguistics (COLING 2008)*.

Voorhees, Ellen. 2001. Overview of trec 2001 question answering track. In *In Proceedings of TREC*.

Voorhees, Ellen. 2002. Overview of trec 2002 question answering track. In *In Proceedings of TREC*.

Voorhees, Ellen. 2003. Overview of trec 2003 question answering track. In *In Proceedings of TREC*.

Zhai, ChengXiang and John D. Lafferty. 2004. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, pages 179–214.

A Data Driven Approach to Query Expansion in Question Answering

Leon Derczynski, Jun Wang, Robert Gaizauskas and Mark A. Greenwood
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello
Sheffield S1 4DP UK
{aca00lad, acp07jw}@shef.ac.uk
{r.gaizauskas, m.greenwood}@dcs.shef.ac.uk

Abstract

Automated answering of natural language questions is an interesting and useful problem to solve. Question answering (QA) systems often perform information retrieval at an initial stage. Information retrieval (IR) performance, provided by engines such as Lucene, places a bound on overall system performance. For example, no answer bearing documents are retrieved at low ranks for almost 40% of questions.

In this paper, answer texts from previous QA evaluations held as part of the Text REtrieval Conferences (TREC) are paired with queries and analysed in an attempt to identify performance-enhancing words. These words are then used to evaluate the performance of a query expansion method.

Data driven extension words were found to help in over 70% of difficult questions. These words can be used to improve and evaluate query expansion methods. Simple blind relevance feedback (RF) was correctly predicted as unlikely to help overall performance, and an possible explanation is provided for its low value in IR for QA.

1 Introduction

The task of supplying an answer to a question, given some background knowledge, is often considered fairly trivial from a human point of view, as long as the question is clear and the answer is known. The aim of an automated question answering system is to provide a single, unambiguous response to a natural language question, given a text collection as a knowledge source, within a certain amount of time. Since 1999, the Text Retrieval Conferences have included a task to evaluate such systems, based on a large pre-defined corpus (such as AQUAINT, containing around a million news articles in English) and a set of unseen questions.

Many information retrieval systems perform document retrieval, giving a list of potentially relevant documents when queried – Google's and Yahoo!'s search products are examples of this type of application. Users formulate a query using a few keywords that represent the task they are trying to perform; for example, one might search for "eiffel tower height" to determine how tall the Eiffel tower is. IR engines then return a set of references to potentially relevant documents.

In contrast, QA systems must return an exact answer to the question. They should be confident that the answer has been correctly selected; it is no longer down to the user to research a set of document references in order to discover the information themselves. Further, the system takes a natural language question as input, instead of a few user-selected key terms.

Once a QA system has been provided with a question, its processing steps can be described in three parts - Question Pre-Processing, Text Retrieval and Answer Extraction:

1. Question Pre-Processing TREC questions are grouped into series which relate to a given target. For example, the target may be "Hindenburg disaster" with questions such as "What type of craft was the Hindenburg?" or "How fast could it travel?". Questions may include pronouns ref-

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 34–41
Manchester, UK. August 2008

erencing the target or even previous answers, and as such require processing before they are suitable for use.

2. Text Retrieval An IR component will return a ranked set of texts, based on query terms. Attempting to understand and extract data from an entire corpus is too resource intensive, and so an IR engine defines a limited subset of the corpus that is likely to contain answers. The question should have been pre-processed correctly for a useful set of texts to be retrieved – including anaphora resolution.

3. Answer Extraction (AE) Given knowledge about the question and a set of texts, the AE system attempts to identify answers. It should be clear that only answers within texts returned by the IR component have any chance of being found.

Reduced performance at any stage will have a knock-on effect, capping the performance of later stages. If questions are left unprocessed and full of pronouns (e.g.,"When did it sink?") the IR component has very little chance of working correctly – in this case, the desired action is to retrieve documents related to the Kursk submarine, which would be impossible.

IR performance with a search engine such as Lucene returns no useful documents for at least 35% of all questions – when looking at the top 20 returned texts. This caps the AE component at 65% question "coverage". We will measure the performance of different IR component configurations, to rule out problems with a default Lucene setup.

For each question, answers are provided in the form of regular expressions that match answer text, and a list of documents containing these answers in a correct context. As references to correct documents are available, it is possible to explore a data-driven approach to query analysis. We determine which questions are hardest then concentrate on identifying helpful terms found in correct documents, with a view to building a system than can automatically extract these helpful terms from unseen questions and supporting corpus. The availability and usefulness of these terms will provide an estimate of performance for query expansion techniques.

There are at least two approaches which could make use of these term sets to perform query expansion. They may occur in terms selected for blind RF (non-blind RF is not applicable to the TREC QA task). It is also possible to build a catalogue of terms known to be useful according to certain question types, thus leading to a dictionary of (known useful) expansions that can be applied to previously unseen questions. We will evaluate and also test blind relevance feedback in IR for QA.

2 Background and Related Work

The performance of an IR system can be quantified in many ways. We choose and define measures pertinent to IR for QA. Work has been done on relevance feedback specific to IR for QA, where it is has usually be found to be unhelpful. We outline the methods used in the past, extend them, and provide and test means of validating QA relevance feedback.

2.1 Measuring QA Performance

This paper uses two principle measures to describe the performance of the IR component. *Coverage* is defined as the proportion of questions where at least one answer bearing text appears in the retrieved set. *Redundancy* is the average number of answer bearing texts retrieved for each question (Roberts and Gaizauskas, 2004).

Both these measures have a fixed limit n on the number of texts retrieved by a search engine for a query. As redundancy counts the number of texts containing correct answers, and not instances of the answer itself, it can never be greater than the number of texts retrieved.

The TREC reference answers provide two ways of finding a correct text, with both a regular expression and a document ID. Lenient hits (retrievals of answer bearing documents) are those where the retrieved text matches the regular expression; strict hits occur when the document ID of the retrieved text matches that declared by TREC as correct *and* the text matches the regular expression. Some documents will match the regular expression but not be deemed as containing a correct answer (this is common with numbers and dates (Baeza-Yates and Ribeiro-Neto, 1999)), in which case a lenient match is found, but not a strict one.

The answer lists as defined by TREC do not include every answer-bearing document – only those returned by previous systems and marked as correct. Thus, false negatives are a risk, and strict measures place an approximate lower bound on the system's actual performance. Similarly, lenient

matches can occur out of context, without a supporting document; performance based on lenient matches can be viewed as an approximate upper bound (Lin and Katz, 2005).

2.2 Relevance Feedback

Relevance feedback is a widely explored technique for query expansion. It is often done using a specific measure to select terms using a limited set of ranked documents of size r; using a larger set will bring term distribution closer to values over the whole corpus, and away from ones in documents relevant to query terms. Techniques are used to identify phrases relevant to a query topic, in order to reduce noise (such as terms with a low corpus frequency that relate to only a single article) and query drift (Roussinov and Fan, 2005; Allan, 1996).

In the context of QA, Pizzato (2006) employs blind RF using the AQUAINT corpus in an attempt to improve performance when answering factoid questions on personal names. This is a similar approach to some content in this paper, though limited to the study of named entities, and does not attempt to examine extensions from the existing answer data.

Monz (2003) finds a negative result when applying blind feedback for QA in TREC 9, 10 and 11, and a neutral result for TREC 7 and 8's ad hoc retrieval tasks. Monz's experiment, using $r = 10$ and standard Rocchio term weighting, also found a further reduction in performance when r was reduced (from 10 to 5). This is an isolated experiment using just one measure on a limited set of questions, with no use of the available answer texts.

Robertson (1992) notes that there are issues when using a whole document for feedback, as opposed to just a single relevant passage; as mentioned in Section 3.1, passage- and document-level retrieval sets must also be compared for their performance at providing feedback. Critically, we will survey the intersection between words known to be helpful and blind RF terms based on initial retrieval, thus showing exactly how likely an RF method is to succeed.

3 Methodology

We first investigated the possibility of an IR-component specific failure leading to impaired coverage by testing a variety of IR engines and configurations. Then, difficult questions were identified, using various performance thresholds. Next, answer bearing texts for these harder questions were checked for words that yielded a performance increase when used for query expansion. After this, we evaluated how likely a RF-based approach was to succeed. Finally, blind RF was applied to the whole question set. IR performance was measured, and terms used for RF compared to those which had proven to be helpful as extension words.

3.1 IR Engines

A QA framework (Greenwood, 2004a) was originally used to construct a QA system based on running a default Lucene installation. As this only covers one IR engine in one configuration, it is prudent to examine alternatives. Other IR engines should be tested, using different configurations. The chosen additional engines were: Indri, based on the mature INQUERY engine and the Lemur toolkit (Allan et al., 2003); and Terrier, a newer engine designed to deal with corpora in the terabyte range and to back applications entered into TREC conferences (Ounis et al., 2005).

We also looked at both passage-level and document-level retrieval. Passages can be defined in a number of ways, such as a sentence, a sliding window of k terms centred on the target term(s), parts of a document of fixed (and equal) lengths, or a paragraph. In this case, the documents in the AQUAINT corpus contain paragraph markers which were used as passage-level boundaries, thus making "passage-level" and "paragraph-level" equivalent in this paper. Passage-level retrieval may be preferable for AE, as the number of potential distracters is somewhat reduced when compared to document-level retrieval (Roberts and Gaizauskas, 2004).

The initial IR component configuration was with Lucene indexing the AQUAINT corpus at passage-level, with a Porter stemmer (Porter, 1980) and an augmented version of the CACM (Jones and van Rijsbergen, 1976) stopword list.

Indri natively supports document-level indexing of TREC format corpora. Passage-level retrieval was done using the paragraph tags defined in the corpus as delimiters; this allows both passage- and document-level retrieval from the same index, according to the query.

All the IR engines were unified to use the Porter

		Coverage		Redundancy	
	Year	Len.	Strict	Len.	Strict
Lucene	2004	0.686	0.636	2.884	1.624
	2005	0.703	0.566	2.780	1.155
	2006	0.665	0.568	2.417	1.181
Indri	2004	0.690	0.554	3.849	1.527
	2005	0.694	0.512	3.908	1.056
	2006	0.691	0.552	3.373	1.152
Terrier	2004	-	-	-	-
	2005	-	-	-	-
	2006	0.638	0.493	2.520	1.000

Table 1: Performance of Lucene, Indri and Terrier at paragraph level, over top 20 documents. This clearly shows the limitations of the engines.

		Coverage		Redundancy	
	Year	Len.	Strict	Len.	Strict
Indri	2004	0.926	0.837	7.841	2.663
	2005	0.935	0.735	7.573	1.969
	2006	0.882	0.741	6.872	1.958
Terrier	2004	0.919	0.806	7.186	2.380
	2005	0.928	0.766	7.620	2.130
	2006	0.983	0.783	6.339	2.067

Table 2: Performance of Indri and Terrier at document level IR over the AQUAINT corpus, with $n = 20$

stemmer and the same CACM-derived stopword list.

The top n documents for each question in the TREC2004, TREC2005 and TREC2006 sets were retrieved using every combination of engine, and configuration[1]. The questions and targets were processed to produce IR queries as per the default configuration for the QA framework. Examining the top 200 documents gave a good compromise between the time taken to run experiments (between 30 and 240 minutes each) and the amount one can mine into the data. Tabulated results are shown in Table 1 and Table 2. Queries have had anaphora resolution performed in the context of their series by the QA framework. AE components begin to fail due to excess noise when presented with over 20 texts, so this value is enough to encompass typical operating parameters and leave space for discovery (Greenwood et al., 2006).

A failure analysis (FA) tool, an early version of which is described by (Sanka, 2005), provided reporting and analysis of IR component performance. In this experiment, it provided high level comparison of all engines, measuring coverage and redundancy as the number of documents retrieved, n, varies. This is measured because a perfect engine will return the most useful documents first, followed by others; thus, coverage will be higher for that engine with low values of n.

3.2 Identification of Difficult Questions

Once the performance of an IR configuration over a question set is known, it's possible to produce a simple report listing redundancy for each question. A performance reporting script accesses the FA tool's database and lists all the questions in a particular set with the strict and lenient redundancy for selected engines and configurations. Engines may use passage- or document-level configurations.

Data on the performance of the three engines is described in Table 2. As can be seen, the coverage with passage-level retrieval (which was often favoured, as the AE component performs best with reduced amounts of text) languishes between 51% and 71%, depending on the measurement method. Failed anaphora resolution may contribute to this figure, though no deficiencies were found upon visual inspection.

Not all documents containing answers are noted, only those checked by the NIST judges (Bilotti et al., 2004). Match judgements are incomplete, leading to the potential generation of false negatives, where a correct answer is found with complete supporting information, but as the information has not been manually flagged, the system will mark this as a failure. Assessment methods are fully detailed in Dang et al. (2006). Factoid performance is still relatively poor, although as only 1.95 documents match per question, this may be an effect of such false negatives (Voorhees and Buckland, 2003). Work has been done into creating synthetic corpora that include exhaustive answer sets (Bilotti, 2004; Tellex et al., 2003; Lin and Katz, 2005), but for the sake of consistency, and easy comparison with both parallel work and prior local results, the TREC judgements will be used to evaluate systems in this paper.

Mean redundancy is also calculated for a number of IR engines. Difficult questions were those for which no answer bearing texts were found by either strict or lenient matches in any of the top n documents, using a variety of engines. As soon as one answer bearing document was found by an engine using any measure, that question was deemed *non-difficult*. Questions with mean redundancy of

[1]Save Terrier / TREC2004 / passage-level retrieval; passage-level retrieval with Terrier was very slow using our configuration, and could not be reliably performed using the same Terrier instance as document-level retrieval.

zero are marked *difficult*, and subjected to further analysis. Reducing the question set to just difficult questions produces a TREC-format file for retesting the IR component.

3.3 Extension of Difficult Questions

The documents deemed relevant by TREC must contain some useful text that can help IR engine performance. Such words should be revealed by a gain in redundancy when used to extend an initially difficult query, usually signified by a change from zero to a non-zero value (signifying that relevant documents have been found where none were before). In an attempt to identify where the useful text is, the relevant documents for each difficult question were retrieved, and passages matching the answer regular expression identified. A script is then used to build a list of terms from each passage, removing words in the question or its target, words that occur in the answer, and stopwords (based on both the indexing stopword list, and a set of stems common within the corpus). In later runs, numbers are also stripped out of the term list, as their value is just as often confusing as useful (Baeza-Yates and Ribeiro-Neto, 1999). Of course, answer terms provide an obvious advantage that would not be reproducible for questions where the answer is unknown, and one of our goals is to help query expansion for unseen questions. This approach may provide insights that will enable appropriate query expansion where answers are not known.

Performance has been measured with both the question followed by an extension (Q+E), as well as the question followed by the target and then extension candidates (Q+T+E). Runs were also executed with just Q and Q+T, to provide non-extended reference performance data points. Addition of the target often leads to gains in performance (Roussinov et al., 2005), and may also aid in cases where anaphora resolution has failed.

Some words are retained, such as titles, as including these can be inferred from question or target terms and they will not unfairly boost redundancy scores; for example, when searching for a "Who" question containing the word "military", one may want to preserve appellations such as "Lt." or "Col.", even if this term appears in the answer.

This filtered list of extensions is then used to create a revised query file, containing the base question (with and without the target suffixed) as well as new questions created by appending a candidate extension word.

Results of retrievals with these new question are loaded into the FA database and a report describing any performance changes is generated. The extension generation process also creates custom answer specifications, which replicate the information found in the answers defined by TREC.

This whole process can be repeated with varying question difficulty thresholds, as well as alternative n values (typically from 5 to 100), different engines, and various question sets.

3.4 Relevance Feedback Performance

Now that we can find the helpful extension words (HEWs) described earlier, we're equipped to evaluate query expansion methods. One simplistic approach could use blind RF to determine candidate extensions, and be considered potentially successful should these words be found in the set of HEWs for a query. For this, term frequencies can be measured given the top r documents retrieved using anaphora-resolved query Q. After stopword and question word removal, frequent terms are appended to Q, which is then re-evaluated. This has been previously attempted for factoid questions (Roussinov et al., 2005) and with a limited range of r values (Monz, 2003) but not validated using a set of data-driven terms.

We investigated how likely term frequency (TF) based RF is to discover HEWs. To do this, the proportion of HEWs that occurred in initially retrieved texts was measured, as well as the proportion of these texts containing at least one HEW. Also, to see how effective an expansion method is, suggested expansion terms can be checked against the HEW list.

We used both the top 5 and the top 50 documents in formulation of extension terms, with TF as a ranking measure; 50 is significantly larger than the optimal number of documents for AE (20), without overly diluting term frequencies.

Problems have been found with using entire documents for RF, as the topic may not be the same throughout the entire discourse (Robertson et al., 1992). Limiting the texts used for RF to paragraphs may reduce noise; both document- and paragraph-level terms should be checked.

	Engine			
Year	Lucene Para	Indri Para	Indri Doc	Terrier Doc
2004	76	72	37	42
2005	87	98	37	35
2006	108	118	59	53

Table 3: Number of difficult questions, as defined by those which have zero redundancy over both strict and lenient measures, at $n = 20$. Questions seem to get harder each year. Document retrieval yields fewer difficult questions, as more text is returned for potential matching.

	Engine		
	Lucene	Indri	Terrier
Paragraph	226	221	-
Document	-	121	109

Table 4: Number of difficult questions in the 2006 task, as defined above, this time with $n = 5$. Questions become harder as fewer chances are given to provide relevant documents.

4 Results

Once we have HEWs, we can determine if these are going to be of significant help when chosen as query extensions. We can also determine if a query expansion method is likely to be fruitful. Blind RF was applied, and assessed using the helpful words list, as well as RF's effect on coverage.

4.1 Difficult Question Analysis

The number of difficult questions found at $n = 20$ is shown in Table 3. Document-level retrieval gave many fewer difficult questions, as the amount of text retrieved gave a higher chance of finding lenient matches. A comparison of strict and lenient matching is in Table 5.

Extensions were then applied to difficult questions, with or without the target. The performance of these extensions is shown in Table 6. Results show a significant proportion (74.4%) of difficult questions can benefit from being extended with non-answer words found in answer bearing texts.

4.2 Applying Relevance Feedback

Identifying HEWs provides a set of words that are useful for evaluating potential expansion terms.

		Match type	
		Strict	Lenient
Year	2004	39	49
	2005	56	66
	2006	53	49

Table 5: Common difficult questions (over all three engines mentioned above) by year and match type; $n = 20$.

Difficult questions used	118
Variations tested	6683
Questions that benefited	87 (74.4%)
Helpful extension words (strict)	4973
Mean helpful words per question	42.144
Mean redundancy increase	3.958

Table 6: Using Terrier Passage / strict matching, retrieving 20 docs, with TREC2006 questions / AQUAINT. Difficult questions are those where no strict matches are found in the top 20 IRT from just one engine.

	2004	2005	2006
HEW found in IRT	4.17%	18.58%	8.94%
IRT containing HEW	10.00%	33.33%	34.29%
RF words in HEW	1.25%	1.67%	5.71%

Table 7: "Helpful extension words": the set of extensions that, when added to the query, move redundancy above zero. $r = 5$, $n = 20$, using Indri at passage level.

Using simple TF based feedback (see Section 3.4), 5 terms were chosen per query. These words had some intersection (see Table 7) with the extension words set, indicating that this RF may lead to performance increases for previously unseen questions. Only a small number of the HEWs occur in the initially retrieved texts (IRTs), although a noticeable proportion of IRTs (up to 34.29%) contain at least one HEW. However, these terms are probably not very frequent in the documents and unlikely to be selected with TF-based blind RF. The mean proportion of RF selected terms that were HEWs was only 2.88%. Blind RF for question answering fails here due to this low proportion. Strict measures are used for evaluation as we are interested in finding documents which were not previously being retrieved rather than changes in the distribution of keywords in IRT.

Document and passage based RF term selection is used, to explore the effect of noise on terms, and document based term selection proved marginally superior. Choosing RF terms from a small set of documents ($r = 5$) was found to be marginally better than choosing from a larger set ($r = 50$). In support of the suggestion that RF would be un-

	r				
	5		50		Baseline
Rank	Doc	Para	Doc	Para	
5	0.253	0.251	0.240	0.179	0.312
10	0.331	0.347	0.331	0.284	0.434
20	0.438	0.444	0.438	0.398	0.553
50	0.583	0.577	0.577	0.552	0.634

Table 8: Coverage (strict) using blind RF. Both document- and paragraph-level retrieval used to determine RF terms.

Question: Who was the nominal leader after the overthrow?	
Target: Pakistani government overthrown in 1999	
Extension word	Redundancy
Kashmir	4
Pakistan	4
Islamabad	2.5
Question: Where did he play in college?	
Target: Warren Moon	
Extension word	Redundancy
NFL	2.5
football	1
Question: Who have commanded the division?	
Target: 82nd Airborne division	
Extension word	Redundancy
Gen	3
Col	2
decimated	2
officer	1

Table 9: Queries with extensions, and their mean redundancy using Indri at document level with $n = 20$. Without extensions, redundancy is zero.

likely to locate HEWs, applying blind RF consistently hampered overall coverage (Table 8).

5 Discussion

HEWs are often found in answer bearing texts, though these are hard to identify through simple TF-based RF. A majority of difficult questions can be made accessible through addition of HEWs present in answer bearing texts, and work to determine a relationship between words found in initial retrieval and these HEWs can lead to coverage increases. HEWs also provide an effective means of evaluating other RF methods, which can be developed into a generic rapid testing tool for query expansion techniques. TF-based RF, while finding some HEWs, is not effective at discovering extensions, and reduces overall IR performance.

There was not a large performance change between engines and configurations. Strict paragraph-level coverage never topped 65%, leaving a significant number of questions where no useful information could be provided for AE.

The original sets of difficult questions for individual engines were small – often less than the 35% suggested when looking at the coverage figures. Possible causes could include:

Difficult questions being defined as those for which average redundancy is zero: This limit may be too low. To remedy this, we could increase the redundancy limit to specify an arbitrary number of difficult questions out of the whole set.

The use of both strict and lenient measures: It is possible to get a lenient match (thus marking a question as non-difficult) when the answer text occurs out of context.

Reducing n from 20 to 5 (Table 4) increased the number of difficult questions produced. From this we can hypothesise that although many search engines are succeeding in returning useful documents (where available), the distribution of these documents over the available ranks is not one that bunches high ranking documents up as those immediately retrieved (unlike a perfect engine; see Section 3.1), but rather suggests a more even distribution of such documents over the returned set.

The number of candidate extension words for queries (even after filtering) is often in the range of hundreds to thousands. Each of these words creates a separate query, and there are two variations, depending on whether the target is included in the search terms or not. Thus, a large number of extended queries need to be executed for each question run. Passage-level retrieval returns less text, which has two advantages: firstly, it reduces the scope for false positives in lenient matching; secondly, it is easier to scan result by eye and determine why the engine selected a result.

Proper nouns are often helpful as extensions. We noticed that these cropped up fairly regularly for some kinds of question (e.g. “Who”). Especially useful were proper nouns associated with locations - for example, adding “Pakistani” to a query containing the word Pakistan lifted redundancy above zero for a question on President Musharraf, as in Table 9. This reconfirms work done by Greenwood (2004b).

6 Conclusion and Future Work

IR engines find some questions very difficult and consistently fail to retrieve useful texts even with high values of n. This behaviour is common over many engines. Paragraph level retrieval seems to give a better idea of which questions are hardest, although the possibility of false negatives is present from answer lists and anaphora resolution.

Relationships exist between query words and helpful words from answer documents (e.g. with a military leadership themes in a query, adding the term “general” or “gen” helps). Identification of HEWs has potential use in query expansion. They could be used to evaluate RF approaches, or associated with question words and used as extensions.

Previous work has ruled out relevance feedback

in particular circumstances using a single ranking measure, though this has not been based on analysis of answer bearing texts. The presence of HEWs in IRT for difficult questions shows that guided RF may work, but this will be difficult to pursue. Blind RF based on term frequencies does not increase IR performance. However, there is an intersection between words in initially retrieved texts and words data driven analysis defines as helpful, showing promise for alternative RF methods (e.g. based on TFIDF). These extension words form a basis for indicating the usefulness of RF and query expansion techniques.

In this paper, we have chosen to explore only one branch of query expansion. An alternative data driven approach would be to build associations between recurrently useful terms given question content. Question texts could be stripped of stopwords and proper nouns, and a list of HEWs associated with each remaining term. To reduce noise, the number of times a particular extension has helped a word would be counted. Given sufficient sample data, this would provide a reference body of HEWs to be used as an aid to query expansion.

References

Allan, J., J. Callan, K. Collins-Thompson, B. Croft, F. Feng, D. Fisher, J. Lafferty, L. Larkey, TN Truong, P. Ogilvie, et al. 2003. The Lemur Toolkit for Language Modeling and Information Retrieval.

Allan, J. 1996. Incremental Relevance Feedback for Information Filtering. In *Research and Development in IR*, pages 270–278.

Baeza-Yates, R. and B. Ribeiro-Neto. 1999. *Modern Information Retrieval*. Addison Wesley.

Bilotti, M.W., B. Katz, and J. Lin. 2004. What Works Better for Question Answering: Stemming or Morphological Query Expansion. *Proc. IR for QA Workshop at SIGIR 2004*.

Bilotti, M.W. 2004. Query Expansion Techniques for Question Answering. Master's thesis, Massachusetts Institute of Technology.

Dang, H.T., J. Lin, and D. Kelly. 2006. Overview of the TREC 2006 QA track. *Proc. 15th Text REtrieval Conf.*.

Greenwood, M.A., M. Stevenson, and R. Gaizauskas. 2006. The University of Sheffield's TREC 2006 Q&A Experiments. In *Proc. 15th Text REtrieval Conference*

Greenwood, M.A. 2004a. AnswerFinder: Question Answering from your Desktop. In *Proc. 7th Annual Colloquium for the UK SIG for Computational Linguistics (CLUK '04)*.

Greenwood, M.A. 2004b. Using Pertainyms to Improve Passage Retrieval for Questions Requesting Information about a Location. In *Proc. Workshop on IR for QA (SIGIR 2004)*.

Jones, K.S. and C.J. van Rijsbergen. 1976. IR Test Collections. *J. of Documentation*, 32(1):59–75.

Lin, J. and B. Katz. 2005. Building a Reusable Test Collection for Question Answering. *J. American Society for Information Science and Technology*.

Monz, C. 2003. From Document Retrieval to Question Answering. *ILLC Dissertation Series 2003*, 4.

Ounis, I., G. Amati, V. Plachouras, B. He, C. Macdonald, and D. Johnson. 2005. Terrier IR Platform. *Proc. 27th European Conf. on IR (ECIR 05), Santiago de Compostela, Spain*, pages 517–519.

Pizzato, L.A., D. Molla, and C. Paris. 2006. Pseudo-Relevance Feedback using Named Entities for Question Answering. *Australasian Language Technology Workshop (ALTW2006)*, pages 83–90.

Porter, M. 1980. An Algorithm for Suffix Stripping Program. *Program*, 14(3):130–137.

Roberts, I and R Gaizauskas. 2004. Evaluating Passage Retrieval Approaches for Question Answering. In *Proc. 26th European Conf. on IR*.

Robertson, S.E., S. Walker, M. Hancock-Beaulieu, A. Gull, and M. Lau. 1992. Okapi at TREC. In *Text REtrieval Conf.*, pages 21–30.

Roussinov, D. and W. Fan. 2005. Discretization Based Learning Approach to Information Retrieval. In *Proc. 2005 Conf. on Human Language Technologies*.

Roussinov, D., M. Chau, E. Filatova, and J.A. Robles-Flores. 2005. Building on Redundancy: Factoid Question Answering, Robust Retrieval and the 'Other'. In *Proc. 14th Text REtrieval Conf.*

Sanka, Atheesh. 2005. Passage Retrieval for Question Answering. Master's thesis, University of Sheffield.

Tellex, S., B. Katz, J. Lin, A. Fernandes, and G. Marton. 2003. Quantitative Evaluation of Passage Retrieval Algorithms for Question Answering. *Proc. 26th Annual Int'l ACM SIGIR Conf. on R&D in IR*, pages 41–47.

Voorhees, E. and L. P. Buckland, editors. 2003. *Proc. 12th Text REtrieval Conference*.

Answer Validation by Information Distance Calculation

Fangtao Li, Xian Zhang, Xiaoyan Zhu
State Key Laboratory on Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University, Beijing 100084, China
zxy-dcs@mail.tsinghua.edu.cn

Abstract

In this paper,an information distance based approach is proposed to perform answer validation for question answering system. To validate an answer candidate, the approach calculates the conditional information distance between the question focus and the candidate under certain condition pattern set. Heuristic methods are designed to extract question focus and generate proper condition patterns from question. General search engines are employed to estimate the Kolmogorov complexity, hence the information distance. Experimental results show that our approach is stable and flexible, and outperforms traditional *tfidf* methods.

1 Introduction

Question answering(QA) system aims at finding exact answers to a natural language question. In order to correctly answer a question, several components are implemented including question classification, passage retrieval, answer candidates generation, answer validation etc. Answer Validation is to decide whether the candidate answers are correct or not, or even to determine the accurate confidence score to them. Most of QA systems employ answer validation as the last step to identify the correct answer. If this component fails, it is impossible to enable the question to be correctly answered.

Automatic techniques for answer validation are of great interest among question answering research. With automatic answer validation, the system will carry out different refinements of its searching criteria to check the relevance of new candidate answers. In addition, since most of QA systems rely on complex architectures and the evaluation of their performances requires a huge amount of work, the automatic assessment of candidates with respect to a given question will speed up both algorithm refinement and testing.

Currently, answer validation is mainly viewed as a classification problem or ranking problem. Different models, such as Support Vector Machine (Shen and Klakow, 2006) and Maximum Entropy Model (Ittycheriah et al., 2001), are used to integrate sophisticated linguistic features to determine the correctness of candidates. The answer validation exercise (Penas et al. , 2007) aims at developing systems able to decide whether the answer is correct or not. They formulate answer validation as a text entailment problem. These approaches are dependent on sophisticated linguistic analysis of syntactic and semantic relations between question and candidates. It is quite expensive to use deep analysis for automatic answer validation, especially in large scale data set. Thus it is appropriate to find an alternative solution to this problem. Here, we just consider the English answer validation task.

This paper proposes a novel approach based on information retrieval on the Web. The answer validation problem is reformulated as distance calculation from an answer candidate to a question. The hypothesis is that, among all candidates, the correct answer has the smallest distance from question. We employ conditional normalized min distance, which is based on Kolmogorov Complexity theory (Li and Vitanyi, 1997), for this task. The distance measures the relevance between question

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 42–49
Manchester, UK. August 2008

focus and candidates conditioned on a surface pattern set. For distance calculation, we first extract the question focus, and then a hierarchical pattern set is automatically constructed as condition. Since Kolmogrov Complexity can be approximated through frequency counts. Two types of search engine "Google" and "Altavista" are used to approximate the distance.

The paper is organized as follows: Section 2 describes related work. The fundamental Kolmogorov Complexity theory is introduced in Section 3. Section 4 presents our proposed answer validation method based on information retrieval. In Section 5, we describe the experiments and discussions. The paper is concluded in Section 6.

2 Related Work

Answer Validation is an emerging topic in Question Answering, where open domain systems are often required to rank huge amounts of answer candidates. This task can be viewed as a classification problem or re-ranking problem.

Early question answering systems focused on employing surface text patterns (Subbotin and Subbotin, 2001) for answer validation. Xu et al. (2003) identified that pattern-based approaches got bad performances due to poor system recall. Some researchers exploited machine learning techniques with rich syntactic or semantic features to measure the similarity between question and answer. Ittycheriah et al. (2001) used Maximum Entropy model to combine rich features and automatically learn feature weights. These features included query expansion features, focus features, named entity features, dependency relation features, pattern features et al. Shen and Klakow (2006) presented three methods, including feature vector, string kernel and tree kernel, to represent surface text features and parse tree features in Support Vector Machines. Ko et al. (2007) proposed a probabilistic graphical model to estimate the probability of correctness for all candidate answers. Four types of features were employed, including knowledge-based features, data-driven features, string distance feature and synonym features.

Started in 2006, the annual Answer Validation Exercise (Penas et al. , 2007) aims to develop systems to decide if the answer to a question is correct or not. The English answer validation task is reformulated as a Text Entailment problem. The triplet, including question, answer and supporting text, is given. The system determines if the supporting text can entail the hypothesis, which is a reformulation from the question and answer. All participants used lexical processing, including lemmatization and part-of speech tagging. Some systems used first order logic representations, performed semantic analysis and took the validation decision with a theorem proof.

The above approaches should process deep syntactic and semantic analysis for either questions or candidate answers. The annotated linguistic resource is hard to acquire for the supervised classification problem. Another alternative solution for answer validation is to exploit the redundancy of large scale data. Eric et al. (2007) developed AskMSR question answering system. They focus on the Web as a gigantic data repository with tremendous redundancy that can be exploited to extract the correct answer. Lin (2007) implemented another Web-based question answering system, named ARANEA, which is used approximate tfidf method for answer validation.

3 Preliminaries

3.1 Kolmogorov complexity

Kolmogorov complexity , or *algorithm entropy* , $K(x)$ of a string x is the length of the shortest binary program to compute x. It defines randomness of an individual string. Kolmogorov complexity has been widely accepted as an information theory for individual objects parallel to that of Shannon's information theory which is defined on an ensemble of objects. It has also found many applications in computer science such as average case analysis of algorithms (Li and Vitanyi, 1997). For a universal Turing machine U, the Kolmogorov complexity of a binary string x condition to another binary string y, $K_U(x|y)$, is the length of the shortest (prefix-free) program for U that outputs x with input y. It has been proved that for different universal Turing machine U', for all x, y

$$K_U(x|y) = K_{U'}(x|y) + C,$$

where the constant C depends only on U'. Thus we simply write $K_U(x|y)$ as $K(x|y)$. Define $K(x) = K(x|\epsilon)$, where ϵ is the empty string. For formal definitions and a comprehensive study of Kolmogorov complexity, see (Li and Vitanyi, 1997).

3.2 Information Distance

Based on the Kolmogovov complexity theory, information distance (Bennett et al., 1998) is a universal distance metric, which has been successfully applied to many applications. The information distance $D(x, y)$ is defined as the length of a shortest binary program which can compute x given y as well as compute y from x. It has been proved that , up to an additive logarithmic term, $D(x, y) = \max\{K(x|y), K(y|x)\}$. The normalized version of $D(x, y)$, called the *normalized information distance(NID)*, is defined as

$$d_{max}(x, y) = \frac{\max\{K(x|y), K(y|x)\}}{\max\{K(x), K(y)\}} \quad (1)$$

Parallel to this, the min distance is proposed in (Zhang et al. , 2007), defined as

$$D_{\min}(x, y) = \min\{K(x|y), K(y|x)\}. \quad (2)$$

And the normalized version is

$$d_{min}(x, y) = \frac{\min\{K(x|y), K(y|x)\}}{\min\{K(x), K(y)\}}. \quad (3)$$

3.3 Conditional Information Distance

Conditional information distance is defined as

$$d_{\max}(x, y|c) = \frac{\max\{K(x|y, c), K(y|x, c)\}}{\max\{K(x|c), K(y|c)\}}, \quad (4)$$

$$d_{\min}(x, y|c) = \frac{\min\{K(x|y, c), K(y|x, c)\}}{\min\{K(x|c), K(y|c)\}}. \quad (5)$$

where c is given in both x to y and y to x computation.

The information distance is proved to be universal (Zhang et al. , 2007), that is, if x and y are "close" under any distance measure, they are "close" under the measure of information distance. However, it is not clear yet how to find out such "closeness" in traditional information distance theory. Now the conditional information distance provides a possible solution.Figure 1 gives a more interpretable explanation: the condition c could map the original concepts x and y into different x_c and y_c, thus the variant "closeness" could be reflected by the distance between x_c and y_c, as shown in Figure1.

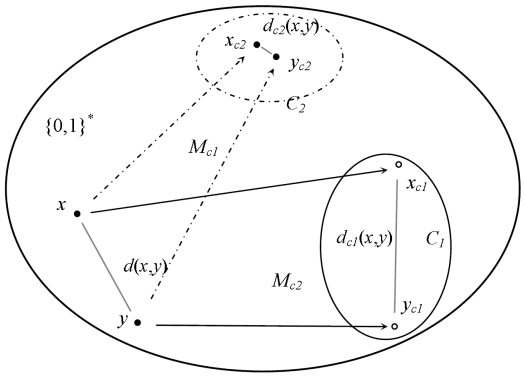

Figure 1: Conditional information distances under different conditions c's

The Kolmogorov complexity is non-computable, that is, to use the information distance measures, we must estimate the K(x) first. There are traditionally two ways to do this: (1) by compression (Li et al. , 2001), and (2) by frequency counting based on coding theorem (Cilibrasi and Vitanyi, 2007). The second approach is implemented in this paper.

4 Answer Validation with Information Distance

Given a question q and a candidate answer c, the answer validation task can be considered as determining the degree of relevance of c with respect to q. The intuition of our approach is that the distance between question and the correct answer is smaller than other candidates. Take the question "What is the capital of the USA?" as an example, among all candidates, the correct answer "Washington" is closest to the question under some distance measure. Thus the answer validation problem is to determine a proper distance measure. Fortunately, it has been proved that the information distance (Bennett et al., 1998) is universal so that the similarity between the question and the answer can surely be discovered using this measure.

Direct calculation of the unconditional distance is difficult and non-flexible. We find it possible and convenient to estimate the conditional information distance between question focus and the answers, under certain context as the condition. As explained previously, different conditions lead to different distance. With the most proper condition and the nearest distance, the best answer can be identified out of previously determined candidates.

The conditional normalized min distance is employed for distance calculation, which is defined

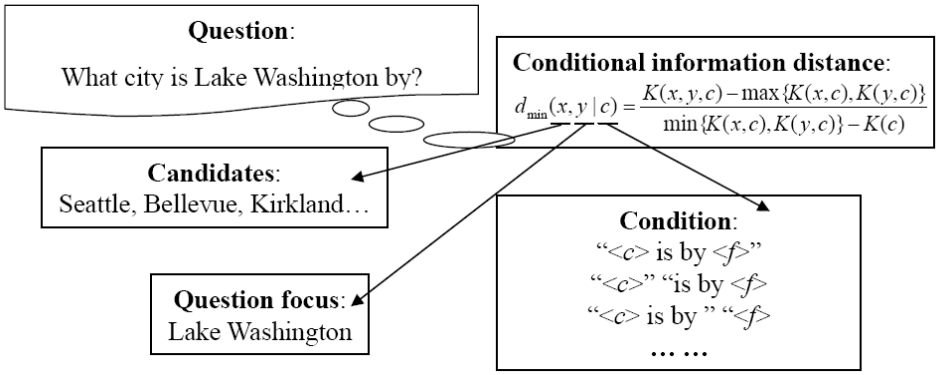

Figure 2: Sample of conditional information distance calculation.

as:

$$d_{min}(x,y|c) = \frac{K\left(c(x,y)\right) - \max\{K\left(c(x,\phi)\right), K\left(c(\phi,y)\right)\}}{\min\{K\left(c(x,\phi)\right), K\left(c(\phi,y)\right)\} - K\left(c(\phi,\phi)\right)}$$

where x represents the answer candidates, y is the question focus, and c is condition pattern. The function $c(x,y)$ will be described in the Distance Calculation section.

Figure 2 shows the procedure of distance calculation. Given a question and a set of candidates, we calculate the min information distance between question focus and candidates conditioned on surface patterns. Obviously, in order to calculate information distance, there are three issues to be addressed:

1. Question Focus Extraction: since the question answer distance is reformulated as the measure between question focus and answer conditioned on the surface pattern, it is important to extract some words or phrases as question focus.

2. Condition Pattern Generation: Obviously, the generation of the condition is the key part. We have built a well revised algorithm, in which proper conditions can be generated from question sentence according to some heuristic rules.

3. Distance Calculation: after question focus and condition patterns are obtained, the last step is calculating the conditional distance to estimate the relevance between question and answer candidates.

4.1 Question Focus Extraction

Most factoid questions refer to specific objects. A question is asked to learn some knowledge for this object from certain perspective. In our approach, we take the key named entity or noun phrase, usually as the subject or the main object of the question sentence as the reference object. Take the question "What city is Lake Washington by" as example, the specific object is "Lake Washington". The question focus is identified using some heuristic rules as follows:

1. The question is processed by shallow parsing. All the noun phrases(NP) are extracted as NP set.

2. All the named entities(NE) in the question are extracted as NE set.

3. If only one same element is identified in both NE and NP set, this element is considered as question focus.

4. If step 3 fails, but two elements from NE and NP set have overlap words, then choose the element with more words as question focus.

5. If step 3 and 4 fail, choose the candidate, which is nearest with verb phrase in dependency tree, as question focus.

4.2 Condition Pattern Generation

A set of hierarchical patterns is automatically constructed for conditional min distance calculation.

4.2.1 Condition Pattern Construction

Several operations are defined for patterns construction from the original question sentence. We describe pattern set construction with a sample question "What year was President Kennedy killed?":

1. With linguistic analysis, the question is split into pieces of tokens. These tokens in-

clude wh-word phrases, preposition phrases, noun phrases, verb phrases, key verb, etc. The example question is split into "What year"(wh-word phrase), "was"(key verb) "President Kennedy" (noun phrases), "killed"(verb phrase).

2. Replace the wh-word phrases with the candidate placeholder $\langle c \rangle$. Then the words "What year" is replaced with placeholder $\langle c \rangle$.

3. Replace the question focus with the focus placeholder $\langle f \rangle$, and add this pattern to the pattern set. The example question focus is identified as "President Kennedy". It is replaced with placeholder $\langle f \rangle$. The first pattern "$\langle c \rangle$ was $\langle f \rangle$ killed?" is generated.

4. Voice Transformation: with morphology techniques, verbs are expanded with all their tense forms (i.e. present, past tense and past participle). The tokens' order is adjusted to transform between active voice and passive voice. Both patterns are added to the patterns set. For sample question, the passive pattern is translated into active pattern, "$\langle c \rangle$ kill $\langle f \rangle$".

5. Preposition addition: for time and location questions, the preposition (i.e. in, on and at) is added before the candidate $\langle c \rangle$; Then the pattern "$\langle c \rangle$ was $\langle f \rangle$ killed" is reformulated as "(in |on) $\langle c \rangle$ was $\langle f \rangle$ killed".

6. Tokens shift: preposition phrase token could be shifted to the begin or the end of pattern, and "key verb" must be shift before the "verb phrase". Then the pattern "(in |on) $\langle c \rangle$ was $\langle f \rangle$ killed" can be reformulated as "$\langle f \rangle$ was killed (in |on) $\langle c \rangle$".

7. Definitional patterns: several heuristic patterns, as introduced at (Hildebrandt et al. , 2004), are added into our final pattern sets, such as "$\langle c \rangle$, $\langle f \rangle$".

By such heuristic rules, the original pattern set is obtained from question sentence. The patterns are initially enclosed in quotation marks, which means exact matching. However, by eliminating these quotations, or reducing the scope that they cover, the matching is relaxed as words co-occurrence. The patterns are expanded into different strict-level patterns by adding or removing quotation marks for each tokens or adjacent tokens combination. Several condition pattern samples are shown in Table 1

Table 1: Sample condition patterns, ' "" ' denotes exact match in web query.

① " <f>(was | were) killed (in | on) <c>"
② " (in | on) <c>, <f>(was | were) killed"
③ " (in | on) <c>" **&** "<f>(was | were) killed"
④ " (in | on) <c>" **&** "<f>" **&** "(was | were) killed"
⑤ in | on <c><f>(was | were) killed

Each operation introduced above is given a predefined confidence coefficient(cc). Then the confidence coefficient of a pattern is defined as the multiplication of cc for all performed operations to generate this pattern.

4.2.2 Condition Pattern Ranking

From the previous step, a set of condition patterns and corresponding confidence coefficient are obtained. Let p_i denotes the ith pattern in the pattern set, and cc_i is the confidence coefficient for the ith pattern. The confidence coefficient estimation in previous section contains much noise. And the patterns with similar confidence coefficient make little difference. Therefore, the exact confidence coefficient value is not directly used. We cluster the patterns into different priority groups. C_j denotes the pattern cluster with jth priority. Here, the smaller j means higher priority. The condition patterns are ranked mainly based on confidence coefficient and the number of double quotation marks. The following algorithm shows each step in detail:

Table 2: patterns ranking algorithm

Input	patterns set $C = \{(p_i, cc_i)\}$
Algorithm	
(1)	Initialize $C_j = \emptyset$, $j = 0$
(2)	if C is empty, end this algorithm
(3)	Select (p_{max}, cc_{max}), where $cc_{max} \geq cc_i, (p_i, cc_i) \in C$
(4)	if C_j is empty, add cc_{max} into C_j, jump to (2)
(5)	select the minimum confidence coefficient (p_{min}, cc_{min}) from C_j, compare it with (p_{max}, cc_{max}). if the number of double quotes("") in p_{min} is equal to the number in p_{max}, add p_{max} into C_j. otherwise, $j = j + 1, C_j = \{p_{max}\}$.
(6)	jump to (2) and repeat

4.3 Distance Calculation

Conditional min distance d_{min} is used to measure the relevance between question and candidate. From section 3, d_{min} is not computable, but approximated by frequency counts based on the coding theory:

$$d_{min}(x,y|c) = \frac{K\left(c(x,y)\right) - \max\{K\left(c(x,\phi)\right), K\left(c(\phi,y)\right)\}}{\min\{K\left(c(x,\phi)\right), K\left(c(\phi,y)\right)\} - K\left(c(\phi,\phi)\right)} = \frac{\log f\left(c(x,y)\right) - \min\{\log f\left(c(x,\phi)\right), \log f\left(c(\phi,y)\right)\}}{\max\{\log f\left(c(x,\phi)\right), \log f\left(c(\phi,y)\right)\} - \log f\left(c(\phi,\phi)\right)}$$

The function $c(x, \emptyset)$ means substituting $\langle c \rangle$ in c by answer candidate x and removing placeholder $\langle f \rangle$ if any. Similar definition applies to $c(y, \emptyset)$, $c(x, y)$. For example, given pattern "$\langle f \rangle$ was invented in $\langle c \rangle$", question focus "the telegraph" and a candidate "1867". $c(x, \emptyset)$ is "was invented in 1867". $c(y, \emptyset)$ is "the telegraph was invented", and $c(x, y)$ is "the telegraph was invented in 1867". The frequency counts $f(x)$ are estimated as the number of returned pages by certain search engine with respect to x . $f(c(\phi, \phi))$ denote the total pages indexed in search engine. Two types of search engines "Google" and "Altavista" are employed.

The patterns are selected in priority order to calculate the information distance for each candidate.

5 Experiment and Discussion

5.1 Experiment Setup

Data set: The standard QA test collection (Lin and Katz, 2006) is employed in our experiments. It consists of 109 factoid questions, covering several domains including history, geography, physics, biology, economics, fashion knowledge, and etc.. 20 candidates are prepared for each questions. All answer candidates are first extracted by the implemented question answering system. Then we review the candidate set for each question. If the correct answer is not in this set, it is manually added into the set.

Performance Metric: The top 1 answer precision and mean reciprocal rank (MRR) are used for performance evaluation.The top 1 answer means the correct answer ranks first with our distance calculation method, and $MRR = \frac{1}{n} * \sum_i (\frac{1}{rank_i})$, in which the $\frac{1}{rank_i}$ is 1 if the correct answer occurs in the first position; 0.5 if it firstly occurs in the second position; 0.33 for the third, 0.25 for the fourth, 0.2 for the fifth and 0 if none of the first five answers is correct.

The open source factoid QA system ARANEA (downloaded from Jimmy Lin's website in 2005) is used for comparison, which implements an approximate $tfidf$ algorithm for candidate scoring. Both ARANEA and our proposed approaches use the internet directly. Google is used as the search engine for ARENEA, and our conditional normalized min distance is calculated with Google and Altavista respectively.

5.2 Experiment Results

The performances of our proposed approach and ARANEA are shown in Table 3. For top 1 answer precision, our conditional min distance calculation method through Google achieves 69.7%, and Altavista is 66.1%, which make 56.6% (69.7% v.s.42.2%) and 50.0% (66.1% v.s 42.2%) improvement compared with ARENEA's $tfidf$ method. Our proposed methods achieve 0.756 and 0.772 compared with ARENEA's 0.581 for MRR measure.

Table 3: Performance comparison, where $d_{min}(G)$ denotes the distance calculation through "Google", $d_{min}(A)$ through "Altavista"

	tfidf	$d_{min}(G)$	$d_{min}(A)$
# of Top 1	46	72	69
% of Top 1	42.2	69.7	66.1
MRR	0.581	0.772	0.756

Table 4 shows some correct answer validation examples. the Google Condition(GC) and the Altavista Condition(AC) columns are the employed condition patterns for distance calculation. For question 1400, the conditional normalized google min distance calculates the distance between question focus "the telegragh" and all 20 answer candidates. The minimum distance score is achieved between "the telegraph" and "1837" with the condition pattern "$\langle f \rangle$ was invented in $\langle c \rangle$". Therefore, the candidate "1837" is validated as the correct answer. Meanwhile, the minimum value for conditional normalized altavista min distance is achieved on the same condition.

These results demonstrate that the distance calculation method provides a feasible solution for answer validation.

In discussion section, we will study three questions:

1. What is the role of search engine?
2. What is the role of condition pattern?
3. What is the role of question focus?

Table 4: Question Examples in conditional information calculation through Google and Altavista. GC:Google Condition; AC:Altavista Condition

ID	Question	GC	AC	Answer	Question focus
1400	When was the telegraph invented?	"?y was invented in ?s"	"?y was invented in ?x"	1837	the telegraph
1401	What is the democratic party symbol?	"?y is ?x"	"?y is ?x"	the donkey	the democratic party symbol
1411	What Spanish explorer discovered the Mississippi River?	"?x discovered ?y"	"?x" "discovered" "?y"	Hernando de Soto	the Mississippi River
1412	Who is the governor of Colorado?	"?y is ?x"	"?y, ?x"	Gov. Bill Ritter	the governor of Colorado
1484	What college did Allen Iverson attend?	"?y attended ?x"	"?x" "did ?y"	Georgetown University	Allen Iverson attend

5.3 Discussions

5.3.1 Role of Search Engine

The rise of world-wide-web has enticed millions of users to create billions of web pages. The redundancy of web information is an important resource for question answering. Our Kolmogorov Complexity based information distance is approximated with query frequency obtained by search engine. Two types of search engines "Google" and "Altavista" are employed in this paper. The number of top 1 correct answer is 72 through "Google" and 69 through "Altavista". There is little difference between two numbers, which shows that the information distance based on Kolmogorov Complexity is independent of special search engine. The performance didn't vary much with the change of search engine. Actually, if the local data is accumulated large enough, the information distance can be approximated without the internet. The quality and size of data set affect the experiment performance.

5.3.2 Role of Condition Pattern

Pattern set offers convenient and flexible condition for information distance calculation. In the experiment, there are 61 questions correctly answered by both Google and Altavista. 46 questions of them employ different patterns. Considering Question 1412, the condition pattern in Google is "$\langle c \rangle$ is $\langle f \rangle$", while in Altavista, it is "$\langle f \rangle$, $\langle c \rangle$". However, the correct answer "Gov. Bill Ritter" is identified by both methods. The information distance is stable over specific condition patterns.

5.3.3 Role of Question Focus

Question focus is considered as the discriminator for the question. The distance between a question and a candidate is reformulated as the distance between question focus and candidate conditioned on a set of surface patterns. The proposed approach may not properly extract the question focus, but the answers can be correctly identified when the condition pattern becomes loose enough. Take the question 1484 "What college did Allen Iverson attend?" as example, the verb "attend" is tagged as "noun", then question focus is mistakenly extracted as "Allen Iverson attend", instead of the correct "Allen Iverson". The two conditional information distance method still identify the correct answer "Georgetown University". Because they both employed the looser condition patterns '"$\langle c \rangle$" "$\langle f \rangle$"' and '"$\langle c \rangle$" did "$\langle f \rangle$"'.Therefore, our proposed distance answer validation methods are robust to the question focus selection component.

From the discussion above, it can be seen that our algorithm is stable and robust, not depending on the specific search engine, condition pattern, and question focus.

6 Conclusions

We have presented a novel approach for answer validation based on information distance. The answer validation task is reformulated as distance calculation between question focus and candidate conditioned on a set of surface patterns. The experiments show that our proposed answer validation method makes a great improvement compared

with ARANEA's *tfidf* method. Furthermore, The experiments show that our approach is stable and robust, not depending on the specific search engine, condition pattern, and question focus. In future work, we will try to calculate information distance in the local constructed data set, and expand this distance measure into other application fields.

Acknowledgement

This work is supported by National Natural Science Foundation of China (60572084, 60621062), Hi-tech Research and Development Program of China (2006AA02Z321), National Basic Research Program of China (2007CB311003).

References

Abraham Ittycheriah, Martin Franz, Wei-Jing Zhu, Adwait Ratnaparkhi, and Richard J.Mammone. 2001. *Question answering using maximum entropy conmponents*. In Proceedings of the Second meeting of the North American Chapter of the Association for Computational Linguistics on Language tecnologies.

Anselmo Penas, A. Rodrigo, F. Verdejo. 2007. *Overview of the Answer Validation Exercise 2007*. Working Notes for the CLEF 2007 Workshop.

C.H. Bennett, P. Gacs, M. Li, P. Vitányi, W. Zurek.. 1998. *Information Distance*. *IEEE Trans. Inform. Theory*, 44:4, 1407–1423.

Eric Brill and Susan Dumais and Michele Banko. 2002. *An analysis of the AskMSR question-answering system*. EMNLP '02: the ACL-02 conference on Empirical methods in natural language processing.

Hildebrandt W., Katz B., and Lin J. 2004. *Answering Definition Questions Using Multiple Knowledge Sources*. Proceedings of Human Language Technology Conference. Boston, USA.

Jeongwoo Ko, Luo Si, Eric Nyberg. 2007. *A Probabilistic Graphical Model for Joint Answer Ranking in Question Answering*. In Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval.

Jimmy Lin and Boris Katz. 2006. *Building a reusable test collection for question answering*. J. Am. Soc. Inf. Sci. Technol..

Jimmy Lin. 2007. *An Exploration of the Principles Underlying Redundancy-Based Factoid Question Answering*. ACM Transactions on Information Systems, 27(2):1-55.

Jinxi Xu, Ana Licuanan and Ralph Weischedel. 2003. *Trec 2003 qa at bbn: Answering definitional questions*. In Proceedings of the 12th Text REtrieval Conference, Gaithersburgh, MD, USA.

Ming Li and Paul MB Vitanyi. 1997. *An Introduction to Kolmogorov Complexity and Its Applications*. Working Notes for the CLEF 2007 Workshop.

M. Li, J. Badger, X. Chen, S. Kwong, P. Kearney, H. Zhang.. 2001. *An information-based sequence distance and its application to whole mitochondrial genome phylogeny*. Bioinformatics, 17:2.

M. Subbotin and S. Subbotin. 2001. *Patterns of Potential Answer Expressions as Clues to the Right Answers*. In TREC-10 Notebook papers. Gaithesburg, MD.

R. Cilibrasi, P.M.B. Vitányi. 2007. *An Exploration of the Principles Underlying Redundancy-Based Factoid Question Answering*. EEE Trans. Knowledge and Data Engineering, 19:3, 370–383.

Shen, Dan and Dietrich Klakow . 2006. *Exploring correlation of dependency relation paths for answer extraction*. In Proceedings of COLING-ACL, Sydney, Australia.

Xian Zhang, Yu Hao, Xiaoyan Zhu, and Ming Li. 2007. *Information Distance from a Question to an Answer*. In Proceedings of the 13th ACM SIGKDD international conference on Knowledge discovery and data mining.

Using lexico-semantic information for query expansion in passage retrieval for question answering

Lonneke van der Plas
LATL
University of Geneva
Switzerland
lonneke.vanderplas@lettres.unige.ch

Jörg Tiedemann
Alfa-Informatica
University of Groningen
The Netherlands
j.tiedemann@rug.nl

Abstract

In this paper we investigate the use of several types of lexico-semantic information for query expansion in the passage retrieval component of our QA system. We have used four corpus-based methods to acquire semantically related words, and we have used one hand-built resource. We evaluate our techniques on the Dutch CLEF QA track.[1] In our experiments expansions that try to bridge the terminological gap between question and document collection do not result in any improvements. However, expansions bridging the knowledge gap show modest improvements.

1 Introduction

Information retrieval (IR) is used in most QA systems to filter out relevant passages from large document collections to narrow down the search for answer extraction modules in a QA system. Accurate IR is crucial for the success of this approach. Answers in paragraphs that have been missed by IR are lost for the entire QA system. Hence, high performance of IR especially in terms of recall is essential. Furthermore, high precision is desirable as IR scores are used for answer extraction heuristics and also to reduce the chance of subsequent extraction errors.

Because the user's formulation of the question is only one of the many possible ways to state the information need that the user might have, there is often a discrepancy between the terminology used by the user and the terminology used in the document collection to describe the same concept. A document might hold the answer to the user's question, but it will not be found due to the TERMINOLOGICAL GAP. Moldovan et al. (2002) show that their system fails to answer many questions (25.7%), because of the terminological gap, i.e. keyword expansion would be desirable but is missing. Query expansion techniques have been developed to bridge this gap.

However, we believe that there is more than just a terminological gap. There is also a KNOWLEDGE GAP. Documents are missed or do not end up high in the ranks, because additional world knowledge is missing. We are not speaking of synonyms here, but words belonging to the same subject field. For example, when a user is looking for information about the explosion of the first atomic bomb, in his/her head a subject field is active that could include: war, disaster, World War II.

We have used three corpus-based methods to acquire semantically related words: the SYNTAX-BASED METHOD, the ALIGNMENT-BASED METHOD, and the PROXIMITY-BASED METHOD. The nature of the relations between words found by the three methods is very different. Ranging from free associations to synonyms. Apart from these resources we have used categorised named entities, such as *Van Gogh* IS-A *painter* and synsets from EWN as candidate expansion terms.

In this paper we have applied several types of lexico-semantic information to the task of query expansion for QA. We hope that the synonyms retrieved automatically, and in particular the synonyms retrieved by the alignment-based method, as these are most precise, will help to overcome the

[1] The Cross-Language Evaluation Forum (http://clef-qa.itc.it/)

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 50–57
Manchester, UK. August 2008

terminological gap. With respect to the knowledge gap, we expect that the proximity-based method would be most helpful as well as the list of categorised named entities. For example, knowing that Monica Seles is a tennis player helps to find relevant passages regarding this tennis star.

2 Related work

There are many ways to expand queries and expansions can be acquired from several sources. For example, one can make use of collection-independent resources, such as EWN. In contrast, collection-dependent knowledge structures are often constructed automatically based on data from the collection.

The results from using collection-independent, hand-built sources are varied. Moldovan et al. (2003) show that using a lexico-semantic feedback loop that feeds lexico-semantic alternations from WordNet as keyword expansions to the retrieval component of their QA system increments the scores by 15%. Also, Pasça and Harabagiu (2001) show substantial improvements when using lexico-semantic information from WordNet for keyword alternation on the morphological, lexical and semantic level. They evaluated their system on question sets of TREC-8 and TREC-9. For TREC-8 they reach a precision score of 55.3% without including any alternations for question keywords, 67.6% if lexical alternations are allowed and 73.7% if both lexical and semantic alternations are allowed.

However, Yang and Chua (2003) report that adding additional terms from WordNet's synsets and glosses adds more noise than information to the query. Also, Voorhees (1993) concludes that expanding by automatically generated synonym sets from EWN can degrade results.

In Yang et al. (2003) the authors use external knowledge extracted from WordNet and the Web to expand queries for QA. Minor improvements are attained when the Web is used to retrieve a list of nearby (one sentence or snippet) non-trivial terms. When WordNet is used to rank the retrieved terms, the improvement is reduced. The best results are reached when structure analysis is added to knowledge from the Web and WordNet. Structure analysis determines the relations that hold between the candidate expansion terms to identify semantic groups. Semantic groups are then connected by conjunction in the Boolean query.

Monz (2003) ran experiments using pseudo relevance feedback for IR in a QA system. The author reports dramatic decreases in performance. He argues that this might be due to the fact that there are usually only a small number of relevant documents. Another reason he gives is the fact that he used the full document to fetch expansion terms and the information that allows one to answer the question is expressed very locally.

A global technique that is most similar to ours uses syntactic context to find suitable terms for query expansion (Grefenstette, 1992; Grefenstette, 1994). The author reports that the gain is modest: 2% when expanded with nearest neighbours found by his system and 5 to 6%, when applying stemming and a second loop of expansions of words that are in the family of the augmented query terms.[2] Although the gain is greater than when using document co-occurrence as context, the results are mixed, with expansions improving some query results and degrading others.

Also, the approach by Qiu and Frei (1993) is a global technique. They automatically construct a similarity thesaurus, based on what documents terms appear in. They use word-by-document matrices, where the features are document IDs, to determine the similarity between words. Expansions are selected based on the similarity to the query concept, i.e. all words in the query together, and not based on the single words in the query independently. The results they get are promising.

Pantel and Ravichandran (2004) have used a method that is not related to query expansion, but yet very related to our work. They have semantically indexed the TREC-2002 IR collection with the ISA-relations found by their system for 179 questions that had an explicit semantic answer type, such as *What* band *was Jerry Garcia with?* They show small gains in performance of the IR output using the semantically indexed collection.

Recent work (Shen and Lapata, 2007; Kaisser and Webber, 2007) that falls outside the scope of this paper, but that is worth mentioning successfully applies semantic roles to question answering.

3 Lexico-semantic information

We have used several types of lexico-semantic information as sources for candidate expansion terms. The first three are automatically acquired

[2] i.e. words that appear in the same documents and that share the first three, four or five letters.

from corpora by means of distributional methods.

- Nearest neighbours from proximity-based distributional similarity
- Nearest neighbours from syntax-based distributional similarity
- Nearest neighbours from alignment-based distributional similarity

The idea behind distributional methods is rooted in the DISTRIBUTIONAL HYPOTHESIS (Harris, 1968). Similar words appear in similar context. The way words are distributed over contexts tells us something about their meaning. Context can be defined in several ways. The way the context is defined determines the type of lexico-semantic knowledge we will retrieve.

For example, one can define the context of a word as the n words surrounding it. In that case proximity to the head word is the determining factor. We refer to these methods that use unstructured context as PROXIMITY-BASED METHODS. The nearest neighbours resulting from such methods are rather unstructured as well. They are merely associations between words, such as *baby* and *cry*. We have used the 80 million-word corpus of Dutch newspaper text (the CLEF corpus) that is also part of the document collection in the QA task to retrieve co-occurrences within sentences.

Another approach is one in which the context of a word is determined by syntactic relations. In this case, the head word is in a syntactic relation to a second word and this second word accompanied by the syntactic relation form the context of the head word. We refer to these methods as SYNTAX-BASED METHODS. We have used several syntactic relations to acquire syntax-based context for our headwords. This method results in nearest neighbours that at least belong to the same semantic and syntactic class, for example *baby* and *son*. We have used 500 million words of newspaper text (the TwNC corpus parsed by Alpino (van Noord, 2006)) of which the CLEF corpus is a subset.

A third method we have used is the ALIGNMENT-BASED METHOD. Here, translations of word, retrieved from the automatic word alignment of parallel corpora are used to determine the similarity between words. This method results in even more tightly related data, as it mainly finds synonyms, such as *infant* and *baby*. We have used the Europarl corpus (Koehn, 2003) to extract word alignments from.[3]

By calculating the similarity between the contexts words are found in, we can retrieve a ranked list of nearest neighbours for any headword. We gathered nearest neighbours for a frequency-controlled list of words, that was still manageable to retrieve. We included all words (nouns, verbs, adjectives and proper names) with a frequency of 150 and higher in the CLEF corpus. This resulted in a ranked list of nearest neighbours for the 2,387 most frequent adjectives, the 5,437 most frequent nouns, the 1,898 most frequent verbs, and the 1,399 most frequent proper names. For all words we retrieved a ranked list of its 100 nearest neighbours with accompanying similarity score.

In addition to the lexico-semantic information resulting from the three distributional methods we used:

- Dutch EuroWordNet (Vossen, 1998)
- Categorised named entities

With respect to the first resource we can be short. We selected the synsets of this hand-built lexico-semantic resource for nouns, verbs, adjectives and proper names.

The categorised named entities are a by-product of the syntax-based distributional method. From the example in (1) we extract the apposition relation between *Van Gogh* and *schilder* 'painter' to determine that the named entity *Van Gogh* belongs to the category of painters.

(1) Van Gogh, de beroemde schilder huurde een atelier, Het Gele huis, in Arles.
'Van Gogh, the famous painter, rented a studio, The Yellow House, in Arles.'

We used the data of the TwNC corpus (500M words) and Dutch Wikipedia (50M words) to extract apposition relations. The data is skewed. The Netherlands appears with 1,251 different labels. To filter out incorrect and highly unlikely labels (often the result of parsing errors) we determined the relative frequency of the combination of the named entity and a category with regard to the frequency of the named entity overall. All categorised named entities with relative frequencies under 0.05

[3] In van der Plas and Tiedemann (2006) there is more information on the syntax-based and alignment-based distributional methods.

Lex. info	Nouns	Adj	Verbs	Proper
Proximity	5.3K	2.4K	1.9K	1.2K
Syntax	5.4K	2.3K	1.9K	1.4K
Align	4.0K	1.2K	1.6K	
Cat. NEs				218K
EWN	44.9K	1.5K	9.0K	1.4K

Table 1: Number of words for which lexico-semantic information is available

were discarded. This cutoff made the number of unwanted labels considerably lower.

In Table 1 we see the amount of information that is contained in individual lexico-semantic resources. It is clear from the numbers that the alignment-based method does not provide nearest neighbours for all head words selected. Only 4.0K nouns from the 5.4K retrieve nearest neighbours. The data is sparse. Also, the alignment-based method does not have any nearest neighbours for proper names, due to decisions we made earlier regarding preprocessing: All words were transformed to lowercase.

The proximity-based method also misses a number of words, but the number is far less important. The amount of information the lists of categorised named entities provide is much larger than the amount of information comprised in the list provided by distributional methods. EWN also provides more information than the distributional methods, except for adjectives.[4]

4 Methodology

In order to test the performance of the various lexico-semantic resources we ran several tests. The baseline is running a standard full-text retrieval engine using Apache Lucene (Jakarta, 2004). Documents have been lemmatised and stop words have been removed.

We applied the nearest neighbours resulting from the three distributional methods as described in section 3. For all methods we selected the top-5 nearest neighbours that had a similarity score of more than 0.2 as expansions.

For EWN all words in the same synset (for all senses) were added as expansions. Since all synonyms are equally similar, we do not have similarity scores for them to be used in a threshold.

The categorised named entities were not only used to expand named entities with the corresponding label, but also to expand nouns with named entities. In the first case all labels were selected. The maximum is not more than 18 labels. In the second case some nouns get many expansions. For example, a noun, such as *vrouw* 'woman', gets 1,751 named entities as expansions. We discarded nouns with more than 50 expansions, as these were deemed too general and hence not very useful.

The last two settings are the same for the expansions resulting from distributional methods and the last two types of lexico-semantic information.

- Expansions were added as root forms
- Expansions were given a weight such that all expansions for one original keyword add up to 0.5.

5 Evaluation

For evaluation we used data collected from the CLEF Dutch QA tracks. The CLEF text collection contains 4 years of newspaper text, approximately 80 million words and Dutch Wikipedia, approximately 50 million words. We used the question sets from the competitions of the Dutch QA track in 2003, 2004, and 2005 (774 in total). Questions in these sets are annotated with valid answers found by the participating teams including IDs of supporting documents in the given text collection. We expanded these list of valid answers where necessary.

We calculated for each run the Mean Reciprocal Rank (MRR).[5] The MRR measures the percentage of passages for which a correct answer was found in the top-k passages returned by the system. The MRR score is the average of 1/R where R is the rank of the first relevant passage computed over the 20 highest ranked passages. Passages retrieved were considered relevant when one of the possible answer strings was found in that passage.

6 Results

In Table 2 the MRR (Mean Reciprocal Rank) is given for the various expansion techniques. Scores are given for expanding the several syntactic categories, where possible. The baseline does not

[4]Note that the number of nouns from EWN is the result of subtracting the proper names.

[5]We used MRR instead of other common evaluation measures because of its stronger correlation with the overall performance of our QA system than, for example, coverage and redundancy (see Tiedemann and Mur (2008)).

MRR					
SynCat	EWN	Syntax	Align	Proxi	Cat.NEs
Nouns	51.52	51.15	51.21	51.38	51.75
Adj	52.33	52.27	52.38	51.71	
Verbs	52.40	52.33	52.21	52.62	
Proper	52.59	50.16		53.94	**55.68**
All	51.65	51.21	51.02	53.36	**55.29**

Table 2: MRR scores for the IR component with query expansion from several sources

#questions (+/-)					
SynCat	EWN	Syntax	Align	Proxi	Cat.NEs
Nouns	27/50	28/61	17/58	64/87	17/37
Adj	3/6	1/2	1/2	31/47	
Verbs	31/51	5/10	8/32	51/56	
Proper	3/2	30/80		76/48	157/106
All	56/94	56/131	25/89	161/147	168/130

Table 3: Number of questions that receive a higher (+) or lower (-) RR when using expansions from several sources

make use of any expansion for any syntactic category and amounts to 52.36.

In Table 3 the number of questions that get a higher and lower reciprocal rank (RR) after applying the individual lexico-semantic resources are given. Apart from expansions on adjectives and proper names from EWN, the impact of the expansion is substantial. The fact that adjectives have so little impact is due to the fact that there are not many adjectives among the query terms.[6]

The negligible impact of the proper names from EWN is surprising since EWN provides more entries for proper names than the proximity-based method (1.2K vs 1.4K, as can be seen in 1). The proximity-based method clearly provides information about proper names that are more relevant for the corpus used for QA, as it is built from a subset of that same corpus. This shows the advantage of using corpus-based methods. The impact of the expansions resulting from the syntax-based method lies in between the two previously mentioned expansions. It uses a corpus of which the corpus used for QA is a subset.

The type of expansions that result from the proximity-based method have a larger effect on the performance of the system than those resulting from the syntax-based method. In Chapter 5 of van der Plas (2008) we explain in greater detail that the proximity-based method uses frequency cutoffs to keep the co-occurrence matrix manageable. The larger impact of the proximity-based nearest neighbours is probably partly due to this decision. The cutoffs for the alignment-based and syntax-based method have been determined after evaluations on EuroWordNet (Vossen, 1998) (see also van der Plas (2008)).

The largest impact results from expanding proper names with categorised named entities. We know from Table 1 in section 3, that this resource has 70 times more data than the proximity-based resource.

For most of the resources the number of questions that show a rise in RR is smaller than the number of questions that receive a lower RR, except for the expansion of proper names by the categorised named entities and the proximity-based method.

The expansions resulting from the syntax-based method do not result in any improvements. As expected, the expansion of proper names from the syntax-based method hurts the performance most. Remember that the nearest neighbours of the syntax-based method often include co-hyponyms. For example, *Germany* would get *The Netherlands* and *France* as nearest neighbours. It does not seem to be a good idea to expand the word *Germany* with other country names when a user, for example, asks the name of the Minister of Foreign Affairs of Germany. However, also the synonyms from EWN and the alignment-based method do not result in improvements.

The categorised named entities provide the most successful lexico-semantic information, when used to expand named entities with their category label. The MRR is augmented by almost 3,5%. It is clear that using the same information in the other direction, i.e. to expand nouns with named entities of the corresponding category hurts the scores. The proximity-based nearest neighbours of proper names raises the MRR scores with 1,5%.

Remember from the introduction that we made a distinction between the terminological gap and the knowledge gap. The lexico-semantic resources that are suited to bridge the terminological gap, such as synonyms from the alignment-based method and EWN, do not result in improvements in the experiments under discussion. However, the lexico-semantic resources that may be used to bridge the knowledge gap, i.e. associations from the proximity-based method and categorised

[6]Moreover, the adjectives related to countries, such as *German* and *French* and their expansion *Germany, France* are handled by a separate list.

CLEF score					
EWN	Syntax	Align	Proxi	Cat.NEs	Baseline
46.3	47.0	46.6	47.6	47.9	46.8

Table 4: CLEF scores of the QA system with query expansion from several sources

named entities, do result in improvements of the IR component.

To determine the effect of query expansion on the QA system as a whole we determined the average CLEF score when using the various types of lexico-semantic information for the IR component. The CLEF score gives the precision of the first (highest ranked) answer only. For EWN, the syntax-based, and the alignment-based nearest neighbours we have used all expansions for all syntactic categories together. For the proximity-based nearest neighbours and the categorised named entities we have limited the expansions to the proper names as these performed rather well.

The positive effect of using categorised named entities and proximity-based nearest neighbours for query expansion is visible in the CLEF scores as well, although less apparent than in the MRR scores from the IR component in Table 2.

6.1 Error analysis

Let us first take a look at the disappointing results regarding the terminological gap, before we move to the more promising results related to the knowledge gap. We expected that the expansions of verbs would be particularly helpful to overcome the terminological gap that is large for verbs, since there is much variation. We will give some examples of expansion from EWN and the alignment-based method.

(2) Wanneer werd het Verdrag van Rome getekend?
'When was the Treaty of Rome signed?'

Align: *teken* 'sign' → typeer 'typify', onderteken 'sign'
EWN: *teken* 'sign' → typeer 'typify', kentekenen 'characterise', kenmerk 'characterise', schilder 'paint', kenschets 'characterise', signeer 'sign', onderteken 'sign', schets 'sketch', karakteriseer 'characterise'.

For the example in (2) both the alignment-based expansions and the expansion from EWN result in a decrease in RR of 0.5. The verb *teken* 'sign' is ambiguous. We see three senses of the verb represented in the EWN list, i.e. drawing, characterising, and signing as in signing an official document. One out of the two expansions for the alignment-based method and 2 out of 9 for EWN are in principle synonyms of *teken* 'sign' in the right sense for this question. However, the documents that hold the answer to this question do not use synonyms for the word *teken*. The expansions only introduce noise.

We found a positive example in (3). The RR score is improved by 0.3 for both the alignment-based expansions and the expansions from EWN, when expanding *explodeer* 'explode' with *ontplof* 'blow up'.

(3) Waar explodeerde de eerste atoombom?
'Where did the first atomic bomb explode?'

Align: *explodeer* 'explode' → ontplof 'blow up'.
EWN: *explodeer* 'explode' → barst los 'burst', ontplof 'blow up', barst uit 'crack', plof 'boom'.

To get an idea of the amount of terminological variation between the questions and the documents, we determined the optimal expansion words for each query, by looking at the words that appear in the relevant documents. When inspecting these, we learned that there is in fact little to be gained by terminological variation. In the 25 questions we inspected we found 1 near-synonym only that improved the scores: *gekke-koeienziekte* 'mad cow disease' → *Creutzfeldt-Jacob-ziekte* 'Creutzfeldt-Jacob disease'.

The fact that we find only few synonyms might be related to a point noted by Mur (2006): Some of the questions in the CLEF track that we use for evaluation look like back formulations.

After inspecting the optimal expansions, we were under the impression that most of the expansions that improved the scores were related to the knowledge gap, rather than the terminological gap. We will now give some examples of good and bad expansions related to the knowledge gap.

The categorised named entities result in the best expansions, followed by the proximity-based expansions. In (4) an example is given for which categorised named entities proved very useful:

(4) Wie is Keith Richard?
'Who is Keith Richard?'

Cat. NEs: *Keith Richard* → gitarist 'guitar player', lid 'member', collega 'colleague', Rolling Stones-gitarist 'Rolling Stones guitar player', Stones-gitarist 'Stones guitar player'.

It is clear that this type of information helps a lot in answering the question in (4). It contains the answer to the question. The RR for this question goes from 0 to 1. We see the same effect for the

question *Wat is NASA?* 'What is NASA?'.

It is a known fact that named entities are an important category for QA. Many questions ask for named entities or facts related to named entities. From these results we can see that adding the appropriate categories to the named entities is useful for IR in QA.

The categorised named entities were not always successful. In (5) we show that the proximity-based expansion proved more helpful in some cases.

(5) Welke bevolkingsgroepen voerden oorlog in Rwanda?
'What populations waged war in Rwanda?'

Proximity: *Rwanda* → Zaïre, Hutu, Tutsi, Ruanda, Rwandees 'Rwandese'.
Cat. NEs: *Rwanda* → bondgenoot 'ally', land 'country', staat 'state', buurland 'neighbouring country'.

In this case the expansions from the proximity-based method are very useful (except for Zaire), since they include the answer to the question. That is not always the case, as can be seen in (6). However, the expansions from the categorised named entities are not very helpful in this case either.

(6) Wanneer werd het Verdrag van Rome getekend?
'When was the treaty of Rome signed?'

Proximity: *Rome* → paus 'pope', Italië, bisschop 'bishop', Italiaans 'Italian', Milaan 'Milan'.
Cat. NEs: *Rome* → provincie 'province', stad 'city', hoofdstad 'capital', gemeente 'municipality'.

IR does identify *Verdrag van Rome* 'Treaty of Rome' as a multi-word term, however it adds the individual parts of multi-word terms as keywords as a form of compound analysis. It might be better to expand the multi-word term only and not its individual parts to decrease ambiguity. *Verdrag van Rome* 'Treaty of Rome' is not found in the proximity-based nearest neighbours, because it does not include multi-word terms.

Still, it is not very helpful to expand the word *Rome* with *pope* for this question that has nothing to do with religious affairs. We can see this as a problem of word sense disambiguation. The association *pope* belongs to Rome in the religious sense, the place where the Catholic Church is seated. Rome is often referred to as the Catholic Church itself, as in *Henry VIII broke from Rome*. Gonzalo et al. (1998) showed in an experiment, where words were manually disambiguated, that a substantial increase in performance is obtained when query words are disambiguated, before they are expanded.

We tried to take care of these ambiguities by using an overlap method. The overlap method selects expansions that were found in the nearest neighbours of more than two query words. Unfortunately, as Navigli and Velardi (2003), who implement a similar technique, using lexico-semantic information from WordNet, note, the COMMON NODES EXPANSION TECHNIQUE works very badly. Also, Voorhees (1993) who uses a similar method to select expansions concludes that the method has the tendency to select very general terms that have more than one sense themselves. In future work we would like to implement the method by Qiu and Frei (1993), as discussed in section 2, that uses a more sophisticated technique to combine the expansions of several words in the query.

7 Conclusion

We can conclude from these experiments on query expansion for passage retrieval that query expansion with synonyms to overcome the terminological gap is not very fruitful. We believe that the noise introduced by ambiguity of the query terms is stronger than the positive effect of adding lexical variants. This is in line with findings by Yang and Chua (2003). On the contrary, Pasça and Harabagiu (2001) were able to improve their QA system by using lexical and semantic alternations from WordNet using feedback loops.

The disappointing results might also be due to the small amount of terminological variation between questions and document collection.

However, adding extra information with regard to the subject field of the query, query expansions that bridge the knowledge gap, proved slightly beneficial. The proximity-based expansions augment the MRR scores with 1.5%. Most successful are the categorised named entities. These expansions were able to augment the MRR scores with nearly 3.5%.

The positive effect of using categorised named entities and proximity-based nearest neighbours for query expansion is visible in the CLEF scores for the QA system overall as well. However, the improvements are less apparent than in the MRR scores from the IR component.

Acknowledgements

This research was carried out in the project *Question Answering using Dependency Relations*, which is part of the research program for *Interactive Multimedia Information eXtraction*, IMIX, financed by NWO, the Dutch Organisation for Scientific Research and partly by the European Community's Seventh Framework Programme (FP7/2007-2013) under grant agreement n 216594 (CLASSIC project: www.classic-project.org).

References

Gonzalo, J., F. Verdejo, I. Chugur, and J. Cigarran. 1998. Indexing with WordNet synsets can improve text retrieval. In *Proceedings of the COLING/ACL Workshop on Usage of WordNet for NLP*.

Grefenstette, G. 1992. Use of syntactic context to produce term association lists for text retrieval. In *Proceedings of the Annual International Conference on Research and Development in Information Retrieval (SIGIR)*.

Grefenstette, G. 1994. *Explorations in automatic thesaurus discovery*. Kluwer Academic Publishers.

Harris, Z. S. 1968. *Mathematical structures of language*. Wiley.

Jakarta, Apache. 2004. Apache Lucene - a high-performance, full-featured text search engine library. http://lucene.apache.org/java/docs/index.html.

Kaisser, M. and B. Webber. 2007. Question answering based on semantic roles. In *Proceedings of de ACL workshop on deep linguistic processing*.

Koehn, P. 2003. Europarl: A multilingual corpus for evaluation of machine translation.

Moldovan, D., M. Passça, S. Harabagiu, and M. Surdeanu. 2002. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics (ACL)*.

Moldovan, D., M. Pasça, S. Harabagiu, and M. Surdeanu. 2003. Performance issues and error analysis in an open-domain question answering system. *ACM Transactions on Information Systems.*, 21(2):133–154.

Monz, C. 2003. *From Document Retrieval to Question Answering*. Ph.D. thesis, University of Amsterdam.

Mur, J. 2006. Increasing the coverage of answer extraction by applying anaphora resolution. In *Fifth Slovenian and First International Language Technologies Conference (IS-LTC)*.

Navigli, R. and P. Velardi. 2003. An analysis of ontology-based query expansion strategies. In *Proceedings of the Workshop on Adaptive Text Extraction and Mining (ATEM), in the 14th European Conference on Machine Learning (ECML 2003)*.

Pantel, P. and D. Ravichandran. 2004. Automatically labeling semantic classes. In *Proceedings of the Conference on Human Language Technology and Empirical Methods in Natural Language Processing (HLT/EMNLP)*.

Pasça, M. and S Harabagiu. 2001. The informative role of wordnet in open-domain question answering. In *Proceedings of the NAACL 2001 Workshop on WordNet and Other Lexical Resources*.

Qiu, Y. and H.P. Frei. 1993. Concept-based query expansion. In *Proceedings of the Annual International Conference on Research and Development in Information Retrieval (SIGIR)*, pages 160–169.

Shen, D. and M. Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of EMNLP*.

Tiedemann, J. and J. Mur. 2008. Simple is best: Experiments with different document segmentation strategies for passage retrieval. In *Proceedings of the Coling workshop Information Retrieval for Question Answering*. To appear.

van der Plas, L. and J. Tiedemann. 2006. Finding synonyms using automatic word alignment and measures of distributional similarity. In *Proceedings of COLING/ACL*.

van der Plas, Lonneke. 2008. *Automatic lexico-semantic acquisition for question answering*. Ph.D. thesis, University of Groningen. To appear.

van Noord, G. 2006. At last parsing is now operational. In *Actes de la 13eme Conference sur le Traitement Automatique des Langues Naturelles*.

Voorhees, E.M. 1993. Query expansion using lexical-semantic relations. In *Proceedings of the Annual International Conference on Research and Development in Information Retrieval (SIGIR)*.

Vossen, P. 1998. EuroWordNet a multilingual database with lexical semantic networks.

Yang, H. and T-S. Chua. 2003. Qualifier: question answering by lexical fabric and external resources. In *Proceedings of the Conference on European Chapter of the Association for Computational Linguistics (EACL)*.

Yang, H., T-S. Chua, Sh. Wang, and Ch-K. Koh. 2003. Structured use of external knowledge for event-based open domain question answering. In *Proceedings of the Annual International Conference on Research and Development in Information Retrieval (SIGIR)*.

Evaluation of Automatically Reformulated Questions in Question Series

Richard Shaw, Ben Solway, Robert Gaizauskas and Mark A. Greenwood
Department of Computer Science
University of Sheffield
Regent Court, 211 Portobello
Sheffield S1 4DP UK
{aca04rcs, aca04bs}@shef.ac.uk
{r.gaizauskas, m.greenwood}@dcs.shef.ac.uk

Abstract

Having gold standards allows us to evaluate new methods and approaches against a common benchmark. In this paper we describe a set of gold standard question reformulations and associated reformulation guidelines that we have created to support research into automatic interpretation of questions in TREC question series, where questions may refer anaphorically to the target of the series or to answers to previous questions. We also assess various string comparison metrics for their utility as evaluation measures of the proximity of an automated system's reformulations to the gold standard. Finally we show how we have used this approach to assess the question processing capability of our own QA system and to pinpoint areas for improvement.

1 Introduction

The development of computational systems which can answer natural language questions using large text collections as knowledge sources is widely seen as both intellectually challenging and practically useful. To stimulate research and development in this area the US National Institute of Standards and Technology (NIST) has organized a shared task evaluation as one track at the annual TExt Retrieval Conference (TREC) since 1999[1]. These evaluations began by considering factoid-type questions only (e.g. *How many calories are there in a Big Mac?*) each of which was asked in isolation to any of the others. However, in an effort to move the challenge towards a long term vision of interactive, dialogue-based question answering to support information analysts (Burger et al., 2002), the track introduced the notion of question targets and related question series in TREC2004 (Voorhees, 2005), and this approach to question presentation has remained central in each of the subsequent TRECs. In this simulated task, questions are grouped into series where each series has a target of a definition associated with it (see Figure 1). Each question in the series asks for some information about the target and there is a final "other" question which is to be interpreted as "Provide any other interesting details about the target that has not already been asked for explicitly". In this way "each series is a (limited) abstraction of an information dialogue in which the user is trying to define the target. The target and earlier questions in a series provide the context for the current question." (Voorhees, 2005).

One consequence of putting questions into series in this way is that questions may not make much sense when removed from the context their series provides. For example, the question *When was he born?* cannot be sensibly interpreted without knowledge of the antecedent of *he* provided by the context (target or prior questions). Interpreting questions in question series, therefore, becomes a critical component within a QA systems. Many QA systems have an initial document retrieval stage that takes the question and derives a query from it which is then passed to a search engine whose task is to retrieve candidate answering bearing documents for processing by the rest of the system. Clearly a question such as *When was he born?* is unlikely to retrieve documents rele-

[1] http://trec.nist.gov/

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 58–65
Manchester, UK. August 2008

	Target 136: Shiite
Q136.1	Who was the first Imam of the Shiite sect of Islam?
Q136.2	Where is his tomb?
Q136.3	What was this persons relationship to the Prophet Mohammad?
Q136.4	Who was the third Imam of Shiite Muslims?
Q136.5	When did he die?

Figure 1: An Example Question Series

vant to answering a question about Kafka's date of birth if passed directly to a search engine. This problem can be addressed in a naive way by simply appending the target to every question. However, this has several disadvantages: (1) in some cases co-reference in a question series is to the answer of a previous question and not to the target, so blindly substituting the target is not appropriate; (2) some approaches to query formulation and to answer extraction from retrieved documents may require syntactically well-formed questions and may be able to take advantage of the extra information, such as syntactic dependencies, provided in a fully de-referenced, syntactically correct question.

Thus, it is helpful in general if systems can automatically interpret a question in context so as to resolve co-references appropriately, and indeed most TREC QA systems do this to at least a limited extent as part of their question pre-processing. Ideally one would like a system to be able to reformulate a question as a human would if they were to re-express the question so as to make it independent of the context of the preceding portion of the question series. To support the development of such systems it would useful if there were a collection of "gold standard" reformulated questions against which systems' outputs could be compared. However, to the best of our knowledge no such resource exists.

In this paper we describe the creation of such a corpus of manually reformulated questions, measures we have investigated for comparing system generated reformulations against the gold standard, and experiments we have carried out comparing our TREC system's automatic question reformulator against the gold standard and insights we have obtained therefrom.

2 The Gold Standard Corpus

Our aim was to take the questions in a TREC question series and re-express them as questions that would naturally be asked by a human asking them as a single, stand-alone question outside the context of the question series. Our intuition was that most adult native speakers would agree on a small number of variant forms these reformulated questions would take. We explored this intuition by having two persons iteratively reformulate some questions independently, compare results and evolve a small set of guidelines for the process.

2.1 Creating the Gold Standard

Ten question sets were randomly selected from sets available at `http://trec.nist.gov/data/qa/t2007_qadata.html`. These were reformulated separately by two people and results compared. From this an initial set of guidelines was drawn up. Using these guidelines another 10 question sets from the TREC 2007 QA set were independently reformulated and then the guidelines refined.

At this point the reformulators' outputs were sufficiently close to each other and the guidelines sufficiently stable that, given limited resources, it was decided reformulation could proceed singly. Using the guidelines, therefore, a further 48 question sets from 2007 were reformulated, where this time each question set was only reformulated by a single person. Each question set contained between 5 and 7 individual questions therefore around 406 questions were reformulated, creating one or more gold standard forms for each question. In total there are approximately 448 individual reformulations, with a maximum number of 3 reformulations for any single question and a mean of 1.103 reformulations per question.

2.2 Guidelines

Using the above method we derived a set of simple guidelines which anyone should be able to follow to create a set of reformulated questions.

Context independence and readability: The reformulation of questions should be understandable outside of the question series context. The reformulation should be written as a native speaker would naturally express it; this means, for example, that stop words are included.

Example: *"How many people were killed 1991*

eruption of Mount Pinatubo?" vs *"How many people were killed in the 1991 eruption of Mount Pinatubo"*. The latter is preferred as it more readable due to the inclusion of stop words *"in the"*.

Reformulate questions so as to maximise search results:

Example: *"Who was William Shakespeare?"* vs *"Who was Shakespeare?"*. *William* should be added to the phrase as it adds extra information which could allow more results to be found.

Target matches a sub-string of the question: If the target string matches a sub-string of the question the target string should substitute the entirety of the substring. Stop-words should not be used when determining if strings and target match but should usually be substituted along with the rest of the target.

Example: Target: *"Sony Pictures Entertainment (SPE)"*; Question: *"What U.S. company did Sony purchase to form SPE?"*; Gold Standard: *"What U.S. company did Sony purchase to form Sony Pictures Entertainment (SPE)?"*

Rephrasing: A Question should not be unnecessarily rephrased.

Example: Target: *"Nissan Corp"*; Question: *"What was Nissan formerly known as?"*; *"What was Nissan Corp. formerly known as?"* is preferred over the other possible reformulation *"Nissan Corp. was formerly known as what?"*.

Previous Questions and Answers: Questions which include a reference to a previous question should be reformulated to include a PREVIOUS_ANSWER variable. Another reformulation should also be provided should a system know it needs the answer to the previous question but has not found one. This should be a reformulation of the previous question within the current question.

Example: Target: *"Harriet Miers withdraws nomination to Supreme Court"*; Question: *"What criterion did this person cite in nominating Miers?"*; Gold Standard 1: *"What criterion did PREVIOUS_ANSWER cite in nominating Harriet Miers?"*; Gold Standard 2: *"What criterion did this person who nominated Harriet Miers for the post cite in nominating Harriet Miers?"*

Targets that contain brackets: Brackets in target should be dealt with in the following way. The full target should be substituted into the question in the correct place as one of the Gold Standards. The target without the bracketed word and with it should also be included in the Gold Standard.

Example: Target: *"Church of Jesus Christ of Latter-day Saints (Mormons)"*; Question:*"Who founded the Church of Jesus Christ of Latter-day Saints?"*; Gold Standard 1: *"Who founded the Church of Jesus Christ of Latter-day Saints (Mormons)?"*; Gold Standard 2: *"Who founded the Church of Jesus Christ of Latter-day Saints?"*; Gold Standard 3 *"Who founded the Mormons?"*

Stemming and Synonyms: Words should not be stemmed and synonyms should not be used unless they are found in the target or the current question series. If they are found then both should be used in the Gold Standard.

Example: Target: *"Chunnel"*; Question:*"How long is the Chunnel?"*; Gold Standard: *"How long is the Chunnel?"*; Incorrect reformulation: *"How long is the Channel Tunnel?"*

As the term "Channel Tunnel" is not referenced in this section or hard-coded into the QA engine it cannot be substituted for "Chunnel", even though doing so may increase the probability of finding the correct answer.

It: The word *it* should be interpreted as referring to either the answer of the previous question of that set or if no answer available to the target itself.

Example:Target: *"1980 Mount St. Helens eruption"*; Question: *"How many people died when it erupted?"*; Gold Standard: *"How many people died when Mt. St. Helens' erupted in 1980?"*

Pronouns (1): If the pronouns *he* or *she* are used within a question and the TARGET is of type 'Person' then substitute the TARGET string for the pronoun. If however the PREVIOUS_ANSWER is of type 'Person' then it should be substituted instead as in this case the natural interpretation of the pronoun is to the answer of the previous question.

Example: Target: *"Jay-Z"*; Question: *"When was he born?"*; Gold Standard: *"When was Jay-Z born?"*

Pronouns (2): If the pronouns *his*/*hers*/*their* are used within a question and the TARGET is of type 'Person' then substitute the TARGET string for the pronoun appending the string "'s" to the end of the substitution. If however the PREVIOUS_ANSWER is of type 'Person' then it should be substituted as the natural interpretation of the pronoun is to the answer of the previous question.

Example: Target: *"Jasper Fforde"*; Question: *"What year was his first book written?"*; Gold Standard: *"What year was Jasper Fforde's first book written?"*

3 Evaluation against the Gold Standard

To assess how close a system's reformulation of a question in a questions series is to the gold standard requires a measure of proximity. Whatever metric we adopt should have the property that reformulations that are closer to our gold standard reformulations get a higher score. The closest possible score is achieved by getting an identical string to that of the gold standard. Following conventional practice we will adopt a metric that gives us a value between 0 and 1, where 1 is highest (i.e. a score of 1 is achieved when the pre-processed reformulation and the gold standard are identical).

Another requirement for the metric is that the ordering of the words in the reformulation is not as important as the content of the reformulation. We assume this because one key use for reformulated questions in the retrieval of candidate answer bearing documents and the presence of key content terms in a reformulation can help to find answers when it is used as a query, regardless of their order Ordering does still need to be taken into account by the metric but it should alter the score less than the content words in the reformulation.

Related to this point, is that we would like reformulations that simply append the target onto the end of the original question to score more highly on average than the original questions on their own, since this is a default strategy followed by many systems that clearly helps in many cases. These requirement can help to guide metric selection.

3.1 Choosing a metric

There are many different systems which attempt to measure string similarity. We considered a variety of tools like ROUGE (Lin, 2004) and METEOR (Lavie and Agarwal, 2007) but decided they were unsuitable for this task. ROUGE and METEOR were developed to compare larger stretches of text – they are usually used to compare paragraphs rather than sentences. We decided developing our own metric would be simpler than trying to adapt one of these existing tools.

To explore candidate similarity measures we created a program which would take as input a list of reformulations to be assessed and a list of gold standard reformulations and compare them to each other using a selection of different string comparison metrics. To find out which of these metrics best scored reformulations in the way which we expected, we created a set of test reformulations to compare against the gold standard reformulations.

Three test data sets were created: one where the reformulation was simply the original question, one where the reformulation included the target appended to the end, and one where the reformualation was identical to the gold standard. The idea here was that the without target question set should score less than the with target question set and the identical target question set should have a score of 1 (the highest possible score).

We then had to choose a set of metrics to test and chose to use metrics from the SimMetrics library as it is an open source extensible library of string similarity and distance metrics [2].

3.2 Assessing Metrics

After running the three input files against the metrics we could see that certain metrics gave a score which matched our requirements more closely than others.

Table 1 shows the metrics used and the mean scores across the data set for the different question sets. A description of each of these metrics can be found in the SimMetrics library.

From these results we can see that certain metrics are not appropriate. SmithWaterman, Jaro and JaroWinkler all do the opposite to what we require them to do in that they score a reformulation without the target higher than one with the target. This could be due to over-emphasis on word ordering. These metrics can therefore be discounted.

Levenshtein, NeedlemanWunch and QGrams-Distance can also be discounted as the difference between With target and Without target is not large enough. It would be difficult to measure improvements in the system if the difference is this small. MongeElkan can also be discounted as overall its scores are too large and for this reason it would be difficult to measure improvements using it.

Of the five remaining metrics – DiceSimilarity, JaccardSimilarity, BlockDistance, EuclideanDistance and CosineSimilarity – we decided that we should discount EuclideanDistance as it had the smallest gap between with target and without target. We now look at the other four metrics in more detail[3]:

[2] `http://www.dcs.shef.ac.uk/~sam/simmetrics.html`

[3] Refer to Manning and Schütze (2001) for more details on these algorithms.

Metric	Without Target	With Target	Identical
JaccardSim.	0.798	0.911	1.0
DiceSim.	0.872	0.948	1.0
CosineSim.	0.878	0.949	1.0
BlockDistance	0.869	0.941	1.0
EuclideanDistance	0.902	0.950	1.0
MongeElkan	0.922	0.993	1.0
Levenshtein	0.811	0.795	1.0
NeedlemanWunch	0.830	0.839	1.0
SmithWaterman	0.915	0.859	1.0
QGramsDistance	0.856	0.908	1.0
JaroWinkler	0.855	0.831	0.993
Jaro	0.644	0.589	0.984

Table 1: Mean scores across the data set for each of the different question sets.

3.2.1 Block Distance

Block Distance metric is variously named block distance, L1 distance or city block distance. It is a vector-based approach, where q and r are defined in n-dimensional vector space. The L_1 or block distance is calculated from summing the edge distances.

$$L_1(q,r) \;=\; \sum_y |\, q(y) - r(y)|$$

This can be described in two dimensions with discrete-valued vectors. When we can picture the set of points within a grid, the distance value is simply the number of edges between points that must be traversed to get from q to r within the grid. This is the same problem as getting from corner a to b in a rectilinear street map, hence the name "city-block metric".

3.2.2 Dice Similarity

This is based on Dice coefficient which is a term based similarity measure (0-1) whereby the similarity measure is defined as twice the number of terms common to compared entities divided by the total number of terms in both. A coefficient result of 1 indicates identical vectors while a 0 indicates orthogonal vectors.

$$Dice\ Coefficient = \frac{2 * |S_1 \cap S_2|}{|S_1| + |S_2|}$$

3.2.3 Jaccard Similarity

This is a token based vector space similarity measure like the cosine distance. Jaccard Similarity uses word sets from the comparison instances to evaluate similarity. The Jaccard measure penalizes a small number of shared entries (as a portion of all non-zero entries) more than the Dice coefficient. Each instance is represented as a Jaccard vector similarity function. The Jaccard similarity between two vectors X and Y is $(X \cdot Y)/(|X||Y| - (X \cdot Y))$ where $(X \cdot Y)$ is the inner product of X and Y, and $|X| = (X \cdot X)^{1/2}$, i.e. the Euclidean norm of X. This can more easily be described as $(|X \cap Y|)/(|X \cup Y|)$

3.2.4 Cosine similarity

This is a common vector based similarity measure similar to the Dice Coefficient. The input string is transformed into vector space so that the Euclidean cosine rule can be used to determine similarity. The cosine similarity is often paired with other approaches to limit the dimensionality of the problem. For instance with simple strings a list of stopwords is used to reduce the dimensionality of the comparison. In theory this problem has as many dimensions as terms exist.

$$\cos(q,r) \;=\; \frac{\sum_y \; q(y)r(y)}{\sqrt{\sum_y q(y)^2}\sum_y r(y)^2}$$

3.3 Using bigrams and trigrams

All four of these measures appear to value the content of the strings higher than ordering which is what we want our metric to do. However the scores are quite large, and as a result we considered refining the metrics to give scores that are not as close to 1. To do this we decided to try and increase the importance of ordering by also taking into account shared bigrams and trigrams. As we do not want ordering to be too important in our metric we introduced a weighting mechanism into the program to

Metric	Without Target	With Target	ΔGap
Dice	0.872	0.948	+0.076
Cosine	0.878	0.949	+0.071
Jaccard	0.798	0.911	+0.113
Block	0.869	0.941	+0.072

Table 2: Results for Unigram weighting

Metric	Without Target	With Target	ΔGap
Dice	0.783	0.814	-3.6
Cosine	0.789	0.816	-3.5
Jaccard	0.698	0.748	-5.5
Block	0.782	0.811	-3.5

Table 3: U:1, B:1, T:0

allow us to used a weighted combination of shared unigrams, bigrams and trigrams.

The results for just unigram weighting is shown in Table 2.

We began by testing the metrics by introducing just bigrams to give us an idea of what effect they would have. A weight ratio of U:1, B:1, T:0 was used (where U:unigram, B:bigram, T:trigram). The results are shown in Table 3.

The Δ Gap column is the increase in the difference between Without Target and With Target from the first test run which used only unigrams.

The introduction of bigrams decreases the gap between Without Target and With Target. It also lowers the scores which is good as it is then easier to distinguish between perfect reformulations and reformulations which are close but not perfect. This means that the introduction of bigrams is always going to decrease a system's ability to distinguish between Without Target and With Target. We had to now find the lowest decrease in this gap whilst still lowering the score of the with target result.

From the results of the bigrams we expected that the introduction of trigrams would further decrease the gap ($U:1, B:1, T:1$). The results proved this and are shown in Table 4.

Metric	Without Target	With Target	ΔGap
Dice	0.725	0.735	-6.4
Cosine	0.730	0.735	-6.3
Jaccard	0.639	0.663	-9.0
Block	0.724	0.733	-6.1

Table 4: U:1, B:1, T:1

The introduction of trigrams has caused the gaps to significantly drop. It has also lowered the scores too much. From this evidence we decided trigrams are not appropriate to use to refine these metrics.

We now had to try and find the best weighting of unigram to bigram that would lower the With Target score from 1.0 whilst still keeping the gap between Without Target and With Target high.

We would expect that further increasing the bigram weighting would further decrease the gap and the With Target score. The results in Table 5 show this to be the case. However this has decreased the gap too much. The next step was to look at decreasing the weighting of the bigrams.

Metric	Without Target	With Target	ΔGap
Dice	0.754	0.770	-4.8
Cosine	0.759	0.771	-4.9
Jaccard	0.664	0.694	-7.4
Block	0.753	0.767	-4.6

Table 5: U:1, B:2

Metric	Without Target	With Target	ΔGap
Dice	0.813	0.859	-2.4
Cosine	0.819	0.860	-2.4
Jaccard	0.731	0.802	-3.7
Block	0.811	0.854	-2.2

Table 6: U:2, B:1

Table 6 shows that the gap has decreased slightly but the With Target score has decreased by around 10% on average. The Jaccard score for this run is particularly good as it has a good gap and is not too close to 1.0. The Without Target is also quite low which is what we want.

$U:2, B:1$ is currently the best weighting found with the best metric being Jaccard. Further work in this area could be directed at further modifying these weightings using machine learning techniques to refine the weightings using linear regression.

4 Our system against the Metric

Our current pre-processing system takes a question and its target and looks to replace pronouns like "he", "she" and certain definite nominals with the target and also to replace parts of the target with the full target (Gaizauskas et al., 2005). Given our choice of metric we would hope that this strat-

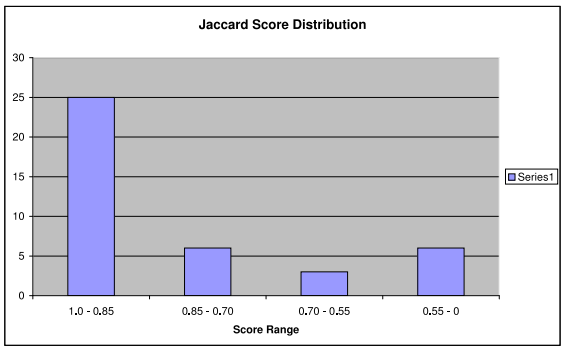

Figure 2: Graph of Jaccard score distribution

egy gets a better score than just adding the target on the end, as the ordering of the words is also taken into account by our pre-processing as it tries to achieve natural reformulations like those of our gold standard. We would therefore expect that it achieves at least the same score as adding the target on the end, which is its default strategy when no co-reference can be determined, though of course incorrect coreference resolutions will have a negative effect. One of the aims of creating the gold standard and a comparison metric was to quickly identify whether strategies such as ours are working and if not where not.

A subset of the gold standard was preprocessed by our system then compared against the results of doing no reformulation and of reformulating by simply appending the target.

Tables 7 and 8 shows how our system did in comparison. Diff shows the difference between WithTarget and Our System. Table 7 is results for weighting $U:1, B:0, T:0$, Table 8 is results for $U:2, B:1, T:0$.

Our system does do better than just adding the target on the end, and this difference is exaggerated (Table 8) when bigrams are taken into account, as expected since this weighting increases the metric's sensitivity to recognising our system's ability to put the target in the correct place.

Mean scores across a data set tell part of the story, but to gain more insight we need to examine the distribution of scores and then, in order to improve the system, we need to look at questions which have a low score and work out what has gone wrong. Figure 2 shows the distribution of Jaccard scores across the test data set. Looking at the scores from the data set using the U:2,B:1,T:0 weighting we find that the minimum Jaccard score was 0.44 and was for the following example:

Metric	Score
Dice	0.574
Cosine	0.578
Jaccard	0.441
Block	0.574

Table 9: Finding Bad Reformulations

Target: *"Hindenburg disaster"*; Question: *"How many of them were killed"*; Our System: *"How many of Hindenburg disaster were killed"*; Gold Standard: *"How many people were killed during the Hindenburg disaster"*.

The results of comparing our system with the gold standard for this question for all four metrics are shown in Table 9.

The problem here is that our system has wrongly replaced the term "them" with the target when in fact its antecedent was in the previous question in the series *How many people were on board?*. Once again the low score has helped us to quickly identify a problem: the system is only interpreting pronouns as references to the target, which is clearly insufficient. Furthermore should the preprocessing system be altered to address a problem like this the gold system and scoring software can be used for regression testing to ensure no previously correct reformulations have been lost.

Another example of a poor scoring reformulation is:

Target: *"Hindenburg disaster"*; Question: *"What type of craft was the Hindenburg"*; Our System: *"What type of craft was the Hindenburg disaster"*; Gold Standard: *"What type of craft was the Hindenburg"*.

For this example Jaccard gave our system reformulation a score of 0.61. The problem here is our system blindly expanded a substring of the target appearing in the question to the full target without recognizing that in this case the substring is not an abbreviated reference to the target (an event) but to an entity that figured in the event.

5 Conclusions and Future Work

In this paper we have presented a Gold Standard for question reformulation and an associated set of guidelines which can be used to reformulate other questions in a similar fashion. We then evaluated metrics which can be used to assess the effectiveness of the reformulations and validated the whole approach by showing how it could be used to help

Metric	Without Target	With Target	Our System	Diff
Dice	0.776	0.901	0.931	+3.1
Cosine	0.786	0.904	0.936	+3.1
Jaccard	0.657	0.834	0.890	+5.5
Block	0.772	0.888	0.920	+4.2

Table 7: How our system compared, U:1,B:0,T:0

Metric	Without Target	With Target	Our System	Diff
Dice	0.702	0.819	0.889	+8.7
Cosine	0.742	0.822	0.893	+9.2
Jaccard	0.616	0.738	0.839	+12.3
Block	0.732	0.812	0.884	+9.1

Table 8: How our system compared, U:2,B:1,T:0

improve the question pre-processing component of a QA system.

Further work will aim to expand the Gold Standard to at least 1000 questions, refining the guidelines as required. The eventual goal is to incorporate the approach into an evaluation tool such that a developer would have a convenient way of evaluating any question reformulation strategy against a large gold standard. Of course one also needs to develop methods for observing and measuring the effect of question reformulation within question pre-processing upon the performance of downstream components in the QA system, such as document retrieval.

References

Burger, J., C. Cardie, V. Chaudhri, R. Gaizauskas, S. Harabagiu, D. Israel, C. Jacquemin, C-Y. Lin, S. Maiorano, G. Miller, D. Moldovan, B. Ogden, J. Prager, E. Riloff, A. Singhal, R. Shrihari, T. Strzalkowski, E. Voorhees, and R. Weischedel. 2002. Issues, tasks and program structures to roadmap research in question & answering (q&a). Technical report. www-nlpir.nist.gov/projects/duc/papers/qa.Roadmap-paper_v2.doc.

Gaizauskas, Robert, Mark A. Greenwood, Mark Hepple, Henk Harkemaa, Horacio Saggion, and Atheesh Sanka. 2005. The University of Sheffield's TREC 2005 Q&A Experiments. In *Proceedings of the 14th Text REtrieval Conference*.

Lavie, Alon and Abhaya Agarwal. 2007. METEOR: An automatic metric for MT evaluation with high levels of correlation with human judgments. In *Proceedings of the Second Workshop on Statistical Machine Translation*, pages 228–231, Prague, Czech Republic, June. Association for Computational Linguistics.

Lin, Chin-Yew. 2004. Rouge: A package for automatic evaluation of summaries. In Marie-Francine Moens, Stan Szpakowicz, editor, *Text Summarization Branches Out: Proceedings of the ACL-04 Workshop*, pages 74–81, Barcelona, Spain, July. Association for Computational Linguistics.

Manning, Christopher D. and Hinrich Schütze. 2001. *Foundations of Statistical Natural Language Processing*. MIT Press.

Voorhees, E. 2005. Overview of the TREC 2004 question answering track. In *Proceedings of the Thirteenth Text Retrieval Conference (TREC 2004)*. NIST Special Publication 500-261.

Topic Indexing and Retrieval for Factoid QA

Kisuh Ahn
School of Informatics
University of Edinburgh
Edinburgh, UK
k.ahn@sms.ed.ac.uk

Bonnie Webber
School of Informatics
University of Edinburgh
Edinburgh, UK
bonnie@inf.ed.ac.uk

Abstract

The method of *Topic Indexing and Retrieval for QA* persented in this paper enables fast and efficent QA for questions with named entity answers. This is achieved by identifying all possible named entity answers in a corpus off-line and gathering all possible evidence for their direct retrieval as answer candidates using standard IR techniques. An evaluation of this method on 377 TREC questions produced a score of 0.342 in Accuracy and 0.413 in Mean Reciprocal Rank (MRR).

1 Introduction

Many textual QA systems use Information Retrieval to retrieve a subset of the documents/passages from the source corpus in order to reduce the amount of text that needs to be investigated in finding the correct answers. This use of Information Retrieval (IR) plays an important role, since it imposes an upper bound on the performance of the entire QA system: Subsequent answer extraction operations cannot make up for the failure of IR to fetch text that contains correct answers. Several techniques have been developed to cut down the amount of text that must be retrieved in order to ensure against the loss of answer material, but processing any text for downstream operations still takes up valuable on-line time.

In this paper, we present a method, *Topic Indexing and Retrieval for QA*, that turns factoid Question Answering into fine-grained Information Retrieval, where answer candidates are directly retrieved instead of documents/passages. The primary claim here is that for simple named entity answers, this can make for fast and accurate retrieval.

2 The Overall Idea

The answers to many factoid questions are named entities — eg, "Who is the president of India?", "Where was Eric Clapton born?", etc. The basic idea of this paper's central method, *Topic Indexing and Retrieval for Question Answering* (or TOQA subsequently), is to extract such expressions off-line from a textual corpus as potential answers and gather evidence that supports their *direct retrieval* as answers to questions using off-the-shelf IR.

Central here is the notion of *topics*. Under this method, any named entities (proper names) found in a corpus are regarded as potential answers. However, named entities are not just treated as words or phrases but as topics with three kinds of information useful for Question Answering.

First, as a locus of information, a topic has *textual content* which talks about this topic. This comprises the set of all sentences from the corpus that mention this topic. Textual content is important because it provides the means to judge the topic's suitability as an answer to a question via textual similarity between the question and some part of the topic's textual content.

Second, a topic has an *ontological type* (or types). This type information is very important for QA because the question requires the answer to be of certain type. A topic must be of the same type (or some compatible type via ISA relation) in order to be considered as an answer candidate. For example, the question, "Who is the president of India?" requires the answer to be of type PERSON (or more specifically, PRESIDENT).

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 66–73
Manchester, UK. August 2008

Finally, a topic has *relations* to other topics. For example, the topic, "Dolly the sheep", is closely related to the topic, "Ian Wilmut". While the precise nature of this relation may vary, the frequent co-occurence of two topics in sentences can be regarded as an evidence that the two are related. Related topics are useful for question answering because they reduce the search space. For example, the answer to the question, e.g. "Who created Dolly the sheep?" can be found among all the topics that are related to the topic contained in the question (or *question topic*), e.g. "Dolly" here.

These three kinds of information are the base material for Question Answering using topics: they provide the means to directly retrieve answers to questions.

3 Preprocessing

This section describes the technical details of how to collect these three kinds of information used for topic based QA, and how to process and store them off-line in order to enable fast and efficient online question answering. The stored material consists of (1) a *Topic Repository*, which stores topics with their variant names and ontological types, (2) a *topic document* collection that stores the textual content of topics, and (3) a set of indices created by indexing the topic document collection for fast and efficient retrieval.

3.1 The Make Up of Topic Repository

The *Topic Repository* stores topics, along their variant names and their ontological types, in hash tables for fast look-up. Building a topic repository requires identifying topics within the given corpus. For this we have used the C&C named entity recogniser (Curran and Clark, 2003), which is run on pos-tagged and chunked documents in the corpus to identify and extract named entities as potential topics. This also identifies the base type of a subset of named entities as PERSON, LOCATION and ORGANISATION. This is stored for later use in building type-separated indices. When a named entity is identified, we first check whether it represents a topic already found in the topic repository. This is done by checking the *topic-name hash table* in the repository, which serves as the main data storage for the variant names of topics.

To resolve a target named entity to the appropriate topic, we use Wikipedia's Redirect table, which contains many common variant names for the same topic. The topic-name hash table is updated accordingly. Hash table entries consist of pairs like ('George Clooney', 1745442), where the name 'George Clooney' is one of the names that belong to the unique topic with the ID number of 1745442. We currently do nothing to disambiguate topics, so different individuals with the same name will all be considered the same topic.

Fine-grained ontological types of topics are identified and stored as well in a separate *topic-type table*. In order to discover fine-grained topic types, the ontology database Yago is used (Suchanek et al., 2007). Yago contains such information for Wikipedia topics, derived by mapping the category information about target topic supplied by a Wikipedia user to the appropriate WordNet concept. (Wikipedia categories are not consistent and uniform, and they are more like tags that characterise a topic rather than strictly classify it.) Using this ontology to look up the type(s) of each topic-type (i.e. the corresponding WordNet concept) and by tracing up the WordNet concept hierarchy, we created a fine-grained, multi-level (with respect to ISA) topic-type hash table for all the topics in the topic repository.

The topic-type hash table not only contains the ontological type of a topic, but also a significant amount of world knowledge typically associated to the topic, due to the nature of Wikipedia categories as descriptive tags. For example, 'Bill Gates' is identified as 'CEO' (a title-role), and 'Pusan' as 'a province of Korea' (geographical knowledge). Such diverse and significant knowledge, as well as the breadth and the depth of the fine types contained in the topic-type hash table, enable a very powerful match between the answer type from a question to that of a candidate topic.

The set of fine-grained answer types used here differs from the set of answer types such as Li and Roth (2002) used elsewhere in that the set is open-ended, and new types can be added for an entity at any time.

The topic repository is used in re-ranking answer candidates by the fine-grained anwer type and for question topic identification, as well as in building topic document collection to be explained next.

3.2 The Topic Document Collection

As noted, the textual content of a topic is the set of all sentences in a corpus that mentions this topic.

(Since anaphora resolution is not yet performed, the sentences that only mention a particular topic anaphorically are missed.) Such set of sentences is assembled into one file per topic. This can then be regarded as a document on its own with the topic name as its title. We henceforth call such a document, a *topic document*. Figure 1 illustrates a topic document for the topic, Dolly the sheep. The topic document collection thus created for all topics identified can be regarded as a reorganised corpus with respect to the original corpus as the Figure **??** illustrates.

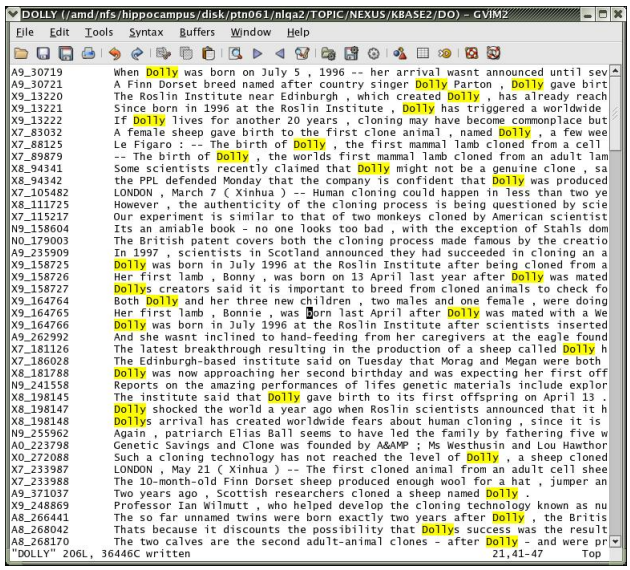

Figure 1: An Example Topic Document: Dolly the sheep

The topic document collection for the full set of topics is a subset of the original corpus, reorganized around topics. The process of creating the topic document collection (which we refer to as the *topic document method*) is actually performed at the same time as the creation of Topic Repository. Any sentence that contains identifiable topics is appended to the topic document of each topic it contains. The topic document collection so created is central to our Question Answering because *retrieving a topic document (specifically, its topic) equates to generating an answer candidate for a given question*. Hence, via topic documents, fine-grained IR can be used to retrieve answers directly.

In order to facilitate such retrieval, however, a topic document collection needs to be indexed. In our implemented system (described in Section 5), this is done using the indexing module of the Lemur Toolkit. For type specific retrieval, three separate indices corresponding to PERSON, LOCATION and ORGANISATION are created according to the base types of the topics identified at the time of Named Entity Recognition. In addition, an index for all topic documents regardless of types, *TOTAL*, is also created for questions from which the answer type cannot be determined or for which their answer types differ from the three base types. Of note here is that separate indices are created only for these base types, as we have not explored separate indexing by fine-grained answer types. These fine-types are only used for reranking after the candidate topics have been retrieved from the base indices.

At the time of retrieval, an appropriate index is to be chosen depending on the answer type identified from the question. This is discussed in the next section.

4 Topic Retrieval and Reranking for QA

The goal is to retrieve a ranked list of topic documents (indicated by their topics) as answers to a given question. In order to do this, the query for the IR operation must be formulated from the question, and the specific answer type must be identified both for retrieval and for any re-ranking of the retrieved list of topics.

Thus, the first necessary operation is *Question Analysis*. Question Analysis identifies the *question type* (eg, definition question, factoid question, list question, etc); the *answer type*, and the *question topics* (if any) and produces a shallow parse of the question text (pos-tagged and chunked) for query formulation. (Identifying the question type is a formality since the method only deals with factoid questions.)

The *question topic identification* is straightforward: Any proper name present in a particular question is a question topic. For *answer type* identification, we use a simple rule based algorithm that looks at the WH-word (e.g. "Where" means location), the head noun of a WH-phrase with "Which" or "What" (e.g. "Which president" means the answer type is of president), and if the main verb is a copula, the head of the post-copula noun phrase (e.g. for "Who is the president ..", here again "president" is the answer type.) WordNet is used to identify the base type of the answer type identified from the question when it is not one of the base types (PERSON, LOCATION, ORGANISATION). For example, "president" is traced to its base type, "PERSON".

Next is the retrieval of topics as answer candidates for a given question. This involves: (1) identifying the appropriate index, (2) formulating the query, and (3) the actual retrieval operation. An appropriate index is chosen based on the base answer type. For example, for the question, "Who is the president of Germany?", the answer type is identified as 'president'. But since the answer type, 'president', is not the base type, WordNet is used to trace from 'president' to a base type (PERSON) and the corresponding index is selected (because separate indices exist only for base types). If none of the three base types is found by this process, the total index is used.

Retrieval uses the InQuery retrieval system within the Lemur Tool Kit (Ogilvie and Callan, 2002). InQuery supports a powerful and flexible *structured query language*. Taking advantage of this, a structured query is formulated from the target question. So for example, the parsed form of the question, "Who is the president of Germany?" is used to generate the following query

\sum(is president of germany
\phrase(president of germany)).

In this example, "president of germany" forms a phrase, and it is inserted as part of the query element with the '\phrase' operator. However, the individual keywords are also included as bag of words since we have found it to give better performance in the trials that we have run. The overall operator is then enclosed by the '\sum' operator that gives the overall score of the query with respect to a document. With this query, search is performed and a ranked list of topics is retrieved. This ranked list is then run through the following operations:

1. Filtering the retrieved list of topics to remove question topics if present.
2. Re-ranking with respect to topic type, preferring the topic that matches the fine answer type.
3. Choosing the highest ranking topic as the answer to the question.

The question topic, in the above example, "Germany", is filtered out if it is found in the list of topics retrieved (using topic-name hash table), which can happen as it is one of the keywords in the query. For the remaining topics in the list, the types of each topic are fetched using the topic-type hash table and matched up to the specific answer type. Re-ranking is performed according to the following rules:

- Topics whose type precisely matches the answer type are ranked higher than any other topics whose types do not precisely match the answer type.
- Topics whose type do not precisely match the answer type but still matches the base type traced from the answer type are ranked higher than any other topics whose types do not match the answer type at all.

Based on these simple rules, the highest-ranking topic is chosen as the answer. Because of the detailed and precise type information stored for each topic, we find this simple procedure works well enough. However, a more sophisticated answer candidate reranking strategy is conceivable based on giving different weights to different degree of match for an answer type.

5 Bi-Topic Indexing

The method described thus far ignores question topics except for filtering them out during post-processing. However, we mentioned in Section 2 that related topics can be exploited in answering questions.

To take advantage of question topics within Topic Indexing and Retrieval, we have adopted the solution of constructing bi-topic documents in contrast to the original topic documents with single topics. An example of a bi-topic document is the following Figure 2, which represents the two topics (Dolly, Ian Wilmut). Such a bi-topic document represents the general relation between two topics via the context in which they co-occur. (As already noted, the precise character of the relation is ignored.) The terms that more frequently appear in such document characterise the relation between the two topics in statistical fashion, and this document would be given a higher score for retrieval with respect to a question, if the question contains such a relatively frequently appearing term. For example, in scoring the bi-topic document pertaining to (Dolly, Ian Wilmut) bi-topic document with respect to the question, "Who created the first cloned sheep, Dolly?", the frequently appearing term in the document, 'cloned' would

give a very high mark for this document with respect to this question.

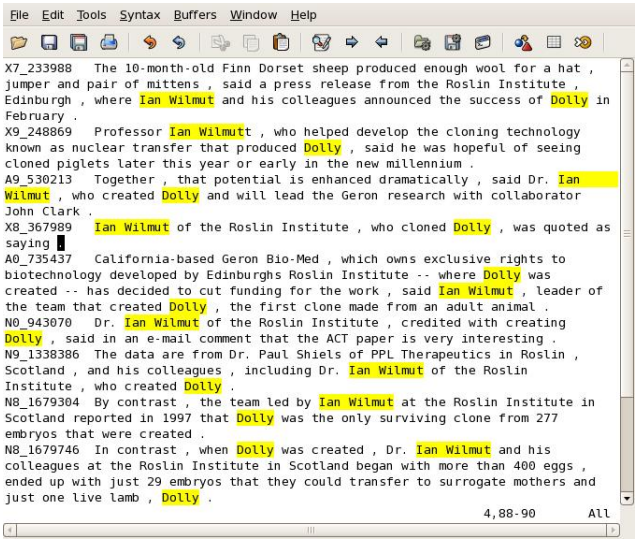

Figure 2: A Bi-Topic Document: (Dolly, Ian Wilmut)

We construct a bi-topic document collection is a recursive application of the topic document method first on the original documents and then to the resulting topic documents. So given a single topic document, e.g. for "Dolly", the same topic document generating process is then applied to this document. This generates a new set of topic documents that, in addition to having their own topics, e.g. "Ian Wilmut", will also contain the topic "Dolly" since the original topic document has the topic "Dolly" in its every sentence. The resulting bi-topic documents would comprise (Dolly, Ian Wilmut), (Dolly, Bonnie), (Dolly, Roslin), etc., all as bi-topic documents. These topic documents all concern the topic "Dolly", which we call the *anchor topic*, and indexing these amounts to creating a "Dolly" (anchored) index. Separate indices for base types as in the case of the single topic documents need not be created since the number of bi-topic documents anchored to one topic is some magnitude smaller compared to the number of total single topic documents.

QA using a bi-topic document index is essentially the same as for the single topic document index, except in selecting the appropriate anchored index using the question topic identified from the question. So the "Dolly" index is chosen if the question topic is "Dolly", as in the question, e.g. "Who created the fist cloned sheep, Dolly?". Reranking based on fine-grained answer types can still be performed although question topic filtering is no longer necessary.

This bi-topic method has the draw-back of generating a lot of documents and corresponding indices since the number of bi-topic documents is the product of the number of topics with all the associated topics. This takes a lot of space for storage and time for generating such a collection. For the evaluation to be described in the next section, we have created bi-topic documents and indices that only pertain to questions (ie only for the question topics within the test set) due to the limitation of space. To be able to scale this method generally, XML information retrieval technique might be applicable as this supports richer retrieval elements other than whole documents and therefore the bi-topic documents pertaining to one anchor topic could be all embedded within one topic document. This is one area we would like to explore further in the future.

The next section characterises and compares the performance of single topic and bi-topic document based methods.

6 Evaluation

6.1 The Evaluation Settings

Evaluation has been carried out to determine whether Topic Indexing and Retrieval using a simple and efficient IR technique for direct answer retrieval can indeed make for an accurate QA system. This has also iluminated those features of the method that contribute to QA performance.

The questions and the corpus (AQUAINT) used for the evaluation are taken from the TREC QA track. 377 questions that have single proper names as answers (ie, excluding list questions, "other" questions and questions without answers) were selected from the TREC 2003/2004/2005 questions. Questions from TREC 2004 and TREC 2005 are grouped around what are called *targets*. A target is basically the question topic, e.g. "When was he born?" where "he" refers to the target, e.g. "Fred Durst". One of the experimental setups takes account of these targets by employing the Bi-topic method discussed in Section 5. This retrieval strategy is also applied to questions from TREC 2003 (that come with no targets), by identifying the question topic in a question and extracting it as a target automatically, in order to see whether it can benefit the QA performance even when the target is not provided manually.

The actual evaluation of the method consists of three experiments, each of which tests a different setting. The common elements for all three are the core answer retrieval system. The aspects that differentiate the three settings are: (1) whether or not a fine-grained answer type is used for reranking, (2) whether single topic documents or bi-topic documents are retrieved.

6.2 The Core Evaluation System

The common core system that implements the answer retrieval method comprises (1) a question analysis module that analyses the question and produces the question type, answer type, the question topics and the shallow parse of the question text and (2) a retrieval module that generates the structured query, selects the appropriate index and retrieves the top 100 topics as answer candidates. This core system performs the basic retrieval operations, to which we add further operations such as answer-type based reranking and target specific retrieval. The addition of some of these features distinguish different setups for the evaluation.

Setup A involves just the core system on single topic document indexing of the AQUAINT corpus, as described in Section 3.2. The resulting topic documents are divided into the three base types (PERSON, LOCATION, ORGANISATION), plus OTHER, as summarised in Table 1. Some examples of entities belong to type OTHER include medicines, roller coasters and software.

KIND	NUM
PERSON	117370
ORGANISATION	67559
LOCATION	48194
OTHER	17942
TOTAL	251065

Table 1: Number of Topic Docs per Types

Setup B is basically the same as setup A, except for the addition of fine-grained answer type re-ranking on the one hundred topics retrieved as answer candidates. That is, elements of this list are re-ranked depending on whether their fine-grained answer type matches the fine-grained answer type identified from the question. Note here that only the coarse answer type (PERSON, LOCATION, ORGANISATION, TOTAL) was used for retrieval, as opposed to the fine-grained type such as PRESIDENT or COMPANY, due to the fact that separate indices exist only for these coarse types. The identification of the fine type of a candidate topic is done by looking up this information in the topic-type hash table as mentioned in Section 3. Again the resulting top candidate is picked as the definite answer.

A@N	A	B	C
1	0.233:88	0.340:128	0.342:129
2	0.316:119	0.406:153	0.443:167
3	0.366:138	0.438:165	0.485:183
4	0.401:151	0.467:176	0.501:189
5	0.430:162	0.491:185	0.515:194
10	0.472:178	0.523:197	0.549:207
15	0.496:187	0.533:201	0.560:211
20	0.512:193	0.541:204	0.560:211
ACC	0.233	0.340	0.342
MRR	0.306	0.395	0.413

Table 2: Results for all setups for all questions

The final setup is setup C. Setup C exploits question topics (targets), as described in Section 5. Targets are explicitly provided in TREC 2004 and TREC 2005 question set. For the TREC 2003, the questions, which do not come with explicit targets, the system automatically extracts a target from the question using a very simple rule: any proper name in the question is regarded as a target. The point of this setup is to test the effectiveness of the bi-topic method discussed in Section 5. The core retrieval procedure is the same as in setup B, except that the index on which the retrieval is performed is selected based on the question topic. In Section 5, we mentioned that a set of indices were built with respect to 'anchor topics'. So the question topic identified from the question (or provided as default) acts as the anchor topic and the index that corresponds to this anchor topic gets chosen. The rest of the process is the same as setup B, and retrieved topics are re-ranked according to the fine-grained answer type.

6.3 Overall Results

Table 2 summarises the results of the experiments across all setups and across all the questions evaluated. The leftmost column indicates the cut-off point (ie, 5 indicates the top-5 answer candidates, 10 indicates the top-10 answer candidates, etc.). The other columns indicate the A@N performance score data for setup A, setup B and setup C respectively at each cut-off point. Each entry comprises

A@N	B-C	C-B	C ∩ B
1	60	61	68
2	61	75	92
3	61	79	104
4	63	76	113
5	66	75	119
10	69	79	128
15	68	78	133
20	69	76	135

Table 3: Overlap between B and C

two scores separated by a colon, representing the ratio of correctly answered questions over all questions and the number of correctly answered questions. The last two rows summarise the results by giving the accuracy (ACC), which is equivalent to the correctness rate at A@1 and the Mean Reciprocal Rank score (MRR).

From this table, it can be seen that both setup B and setup C produced results that are superior to setup A in all measures: accuracy, A@N (for N up to 20) and MRR.

In order to verify whether the differences in scores indicate statistical significance, we have performed *Wilcoxon Matched Signed Rank Test* (Wilcoxon, 1945) on the test data (the differences in ranks for all the questions between setups). This test is suited for testing two related samples when an underlying distribution cannot be assumed (unlike t-test) as with the data here. The statistical test shows that the difference between setup B and setup A is indeed significant ($p = 1.763e - 08$, for P threshold at 0.05) and that the difference between setup C and setup A is also significant ($p = 4.244e-08$). So setup B and setup C perform significantly better than setup A.

Setup C performs slightly better than setup B, both in accuracy (0.342 vs. 0.340) and in MRR (0.413 vs 0.395), but the statistical test shows, this difference is not statistically significant ($p = 0.5729$). However, as the Table 3 shows, setup B and setup C correctly answered different questions. (Setup B answered most of the questions that were correctly answered by setup A, as well as questions that were not correctly answered by setup A). Thus, a further investigation is needed to understand performance differences between setups B and C.

The execution time for each question takes less than one second for both single-topic and bi-topic document indices based retrieval on a single CPU (P4 3.2 Mhz HT) with 512 MB of memory, and the reranking operation did not add any significant amount of time to it.

7 Related Work

In this section, we discuss some of the works on novel indexing techniques for QA that relate to this work.

In predictive annotation (Prager et al., 1999), the text of the target corpus is pre-processed in such a way that phrases that are potential answers are marked and annotated with respect to their answer types (or QA-tokens as they call them) including PERSON$[1], DURATION$, etc. Then the text is indexed not only with ordinary terms but also with these QA-tokens as indexing elements. The main advantage of this approach is that QA-tokens are used as part of the query enhancing the passage retrieval performance. Our work in this paper uses the same predictive annotation technique but differs in that the named entities are indexed as topics and are retrieved directly as answer candidates.

Similar to our approach, Kim et al. (2001) applies predictive annotation method to retrieve answers directly rather than supporting text. For every potential answer in the corpus, a set of text spans up to three sentences long (the sentence in which it appears, plus whatever following sentences that are linked to this sentence via lexical chain totalling no more than three sentences in size) is stored and later sued to retrieve a potential answer. Although similar to our work, the main difference is in the way the textual evidence is aggregated. In Topic Indexing and Retrieval, all the evidence (aka textual content) available throughout the corpus for a possible answer is aggregated, whereas Kim uses text spans up three sentences long from a single document connected by a coreference chain for each answer candidate. Also, topic relations are not exploited as in our work (via Bi-topic documents).

Fleischman et al. (2002) also retrieves answers directly. In what they call *the answer repository approach to Question Answering*, highly precise relational information is extracted from the text collection using text mining techniques based on part of speech patterns. The extracted concept-instance pairs of person name-title such as (Bill

[1] In their notation, the Dollar sign at the end indicates that this is a QA token rather than a term.

Gates, Chairman of Microsoft) are used either solely or in conjunction with a common QA system in producing answers. (Jijkoun et al. (2004) follows a similar approach.) This basically Information Extraction approach taken here can complement our own work for the benefit of increased precision for select types of questions.

In Clifton and Teahan (2004), their knowledge framework based QA system, QITEKAT, prestores possible answers along with their corresponding question templates based on manual and automatic regular expression patterns. That the potential questions are stored as well the answers make this approach different from our approach.

The bi-topic method in this paper has some similarity to Katz and Lin (2000). Here, ternary relations are extracted off-line using manually constructed regular expression patterns on a target text and stored in a database for the use in Question Answering such as in the *START* QA system (Katz et al., 2002). With bi-topic documents in this paper, instead of the precise relations between the two topics, the aggregate context between two particular topics are captured by assembling all statements that mention these two topics together in one file. While this does not give the exact characteristics of the relations involved, it does give some statistical characterization between the two topics to the benefit for QA.

8 Conclusion

In this paper, we have presented the method of *Topic Indexing and Retrieval for QA*. The method effectively turns document retrieval of IR into direct answer retrieval by indexing potential answers (topics) via topic documents. We claimed that the method can be applied in answering simple named-entity questions. The evaluation results indeed show that the method is effective for this type of question, with MRR of 0.413 and accuracy of 0.342 (best run: setup C).

References

Clifton, Terence and William Teahan. 2004. Bangor at TREC 2004: Question answering track. In *Proceedings TREC 2004*.

Curran, J. and S. Clark. 2003. Language independent ner using a maximum entropy tagger. In *Proceedings of the Seventh Conference on Natual Language Learning (CoNLL-03)*, pages 164–167.

Fleischman, Michael, Eduard Hovy, and Abdessamad Echihabi. 2002. Offline strategies for online question answering: Answering questions before they are asked. In *Proceedings of the 41th Annual Meeting of the Association for Computational Linguistics (ACL-2003)*.

Jijkoun, Valentin, Maarten de Rijke, and Jori Mur. 2004. Information extraction for question answering: improving recall through syntactic patterns. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 1284, Morristown, NJ, USA. Association for Computational Linguistics.

Katz, Boris and Jimmy Lin. 2000. REXTOR: A system for generating relations from natural language. In *Proceedings of the ACL 2000 Workshop on Recent Advances in NLP and IR*.

Katz, B., S. Felshin, D. Yuret, A. Ibrahim, J. Lin, G. Marton, A. McFarland, and B. Temelkuran. 2002. Omnibase: Uniform access to heterogeneous data for question answering.

Kim, Harksoo, Kyungsun Kim, Gary Geunbae Lee, and Jungyun Seo. 2001. MAYA: A fast question-answering system based on a predictive answer indexer. In *Proceedings of the 39th Annual Meeting of the Association for Computational Linguistics (ACL-2001) Workshop on Open-Domain Question Answering*.

Li, X. and D. Roth. 2002. Learning question classifiers. In *Proceeding of the 19th International Conference on Computational Linguistics (COLING'02)*.

Ogilvie, P. and J. Callan. 2002. Experiments using the lemur toolkit. In *Proceeding of the 2001 Text Retrieval Conference (TREC 2001)*, pages 103–108.

Prager, John, Dragomir Radev, Eric Brown, Anni Coden, and Valerie Samn. 1999. The use of predictive annotation for question answering in TREC8. In *Proceedings of the Eighth Text REtrieval Conference (TREC-8)*.

Suchanek, Fabian M., Gjergji Kasneci, and Gerhard Weikum. 2007. Yago: A core of semantic knowledge - unifying WordNet and Wikipedia. In Williamson, Carey L., Mary Ellen Zurko, and Prashant J. Patel-Schneider, Peter F. Shenoy, editors, *16th International World Wide Web Conference (WWW 2007)*, pages 697–706, Banff, Canada. ACM.

Wilcoxon, F. 1945. Individual comparisons by ranking methods. *Biometrics*, (1):80–83.

Indexing on Semantic Roles for Question Answering

Luiz Augusto Pizzato
Centre for Language Technology
Macquarie University
Sydney, Australia
pizzato@ics.mq.edu.au

Diego Mollá
Centre for Language Technology
Macquarie University
Sydney, Australia
diego@ics.mq.edu.au

Abstract

Semantic Role Labeling (SRL) has been used successfully in several stages of automated Question Answering (QA) systems but its inherent slow procedures make it difficult to use at the indexing stage of the document retrieval component. In this paper we confirm the intuition that SRL at indexing stage improves the performance of QA and propose a simplified technique named the Question Prediction Language Model (QPLM), which provides similar information with a much lower cost. The methods were tested on four different QA systems and the results suggest that QPLM can be used as a good compromise between speed and accuracy.

1 Introduction

Semantic Role Labeling (SRL) has been implemented or suggested as a means to aid several Natural Language Processing (NLP) tasks such as information extraction (Kogan et al., 2005), multi-document summarization (Barzilay et al., 1999) and machine translation (Quantz and Schmitz, 1994). Question Answering (QA) is one task that takes advantage of SRL, and in fact much of the research about the application of SRL to NLP is related to QA. Thus, Narayanan and Harabagiu (2004) apply the argument-predicate relationship from PropBank (Palmer et al., 2005) together with the semantic frames from FrameNet (Baker et al., 1998) to create an inference mechanism to improve QA. Kaisser and Webber (2007) apply semantic relational information in order to transform questions into information retrieval queries and further analyze the results to find the answers for natural language questions. Sun et al. (2005) use a shallow semantic parser to create semantic roles in order to match questions and answers. Shen and Lapata (2007) developed an answer extraction module that incorporates FrameNet style semantic role information. They deal with the semantic role assignment as a optimization problem in a bipartite graph and the answer extraction as a graph matching over the semantic relations.

Most of the studies that use SRL or similar techniques to QA apply semantic relation tools on the input or output of the Information Retrieval phase of their system. Our paper investigates the use of semantic information for *indexing* documents. Our hypothesis is that allowing Semantic Role information at the indexing stage the question analyzer and subsequent stages of the QA system can obtain higher accuracy by providing an implicit query analyzer as well as more precise retrieval. Theoretically, the inclusion of this information at indexing time can also speed up the overall QA process since syntactic rephrasing or re-ranking of documents based on semantic roles would not be necessary. However, SRL techniques are still highly complex and they demand a computational power that is not yet available to most research groups when working with large corpora. In our experience the annotation of a 3GB corpus, such as the AQUAINT (Graff, 2002), using a semantic role labeler, for instance SwiRL from Surdeanu and Turmo (2005) can take more than one year using a standard PC configuration[1].

In order to efficiently process a corpus with se-

[1] Intel(R) Pentium(R) 4 HT 2.80GHz with 2.0 GB RAM

Coling 2008: Proceedings of the 2nd workshop on Information Retrieval for Question Answering (IR4QA), pages 74–81
Manchester, UK. August 2008

mantic relations, we have developed an alternative annotation strategy based on word-to-word relations instead of noun phrase-to-predicate relations. We define semantic triples based on syntactic clues; this approach was also studied by Litkowski (1999) but some major differences with our work are that we use automatically learned rules to generate the semantic relations, and that we use different semantic labels than those defined by Litkowski, some more specific and some more general. Our annotation scheme is named the Question Prediction Language Model (QPLM) and represents relations between pairs of words using labels such as *Who* and *When*, according to how one word complements the other.

In the following section we provide an overview of the proposed semantic annotation module. Then in Section 3 we detail the information retrieval framework used that allows the indexing and retrieval of semantic information. Section 4 describes the experimental setup and presents the results. Finally, Section 5 presents the concluding remarks and some discussion of further work.

2 Question Prediction Language Model

QPLM, as described in Pizzato and Mollá (2007), represents sentences by specifying the semantic relationship among its components using question words. In this way, we focus on dividing the problem of representing a large sentence into small questions that could be asked about its components. QPLM is expressed by triples $\theta(\omega) \rightarrow \alpha$ where θ is a question word, ω is the word that concerns the question word θ and α is the word that answers the relation θ about ω. For instance the relation $Who(eat) \rightarrow Jack$ tells us that the person who eats is Jack. The representation of our semantic relations as triples $\theta(\omega) \rightarrow \alpha$ is important because it allows the representation of sentences as directed graphs of semantic relations. This representation has the capacity of generating questions about the sentence being analyzed. Figure 1 shows such a representation of the sentence: "*John asked that a flag be placed in every school*".

Having the sentence of Figure 1 and removing a possible answer α from any relation triple, it is possible to formulate a complete question about this sentence that would require α as an answer. For instance, we can observe that removing the node $John$ we obtain the question "*Who asked for a flag to be placed in every school?*" where *Who* was extracted from the triple $Who(ask) \rightarrow John$. The same is valid for other relations, such as removing the word $school$ to obtain the question "*Where did John ask for a flag to be placed?*". The name Question Prediction for this model is due to its capability of generating questions regarding the sentence that has been modeled.

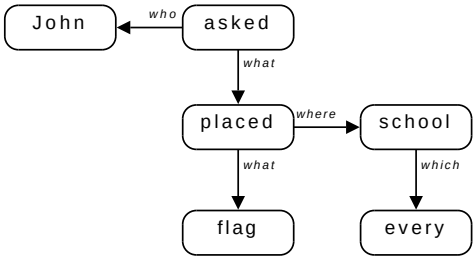

Figure 1: Graph Representation

We have developed a process to automatically annotate QPLM information, the process is rule based where the rules are automatically learned from a corpus obtained from mapping PropBank into QPLM instances. The mapping between semantic roles and QPLM is not one-to-one, which reduces the accuracy of the training corpus. A sample of 40 randomly selected documents was manually evaluated showing that nearly 90% of the QPLM triples obtained were correctly converted from the PropBank mapping. PropBank does not give us some relations that we wish to include such as ownership ($Whose(car) \rightarrow Maria$) or quantity ($HowMany(country) \rightarrow twenty$)), but it does give us the benefits of a large training set covering a variety of different predicates.

Our QPLM annotation tool, like most SRL tools, makes use of a syntactic parser and a named-entity (NE) recognizer. We are currently using Connexor[2] for syntactic parsing and LingPipe[3] for named-entity recognition.

An evaluation of our QPLM annotation has shown a reasonable precision (50%) with a low recall (24%). Both precision and recall seem to be connected with the choice of training corpus. The high precision is influenced by the large training set and the different variety of predicates. The low recall is due to the low amount of connections that can be mapped from one sentence in PropBank to QPLM. As we will present in Section 4, QPLM helps to improve results for QA even when

[2] http://www.connexor.com
[3] http://alias-i.com/lingpipe/

this training corpus is not optimal. This suggests that if a more suitable corpus is used to create the QPLM rules then we can improve the already positive results. An ideal training corpus would contain all QPLM pairs; not only verbs and head of noun phrases but also connections among all relevant words in a sentence.

3 Indexing and Retrieving Semantic Information

A document index that contains information about semantic relations provides a way of finding documents on the basis of meaningful relations among words instead of simply their co-occurrence or proximity to each other. A semantic relation index allows the retrieval of the same piece of information when queried using syntactic variations of the same query such as: "*Bill kicked the ball*" or "*The ball was kicked by Bill*".

Several strategies can be used to build the indexing structure that includes relational information. The task of IR requires fast indexing and retrieval of information regardless of the amount of data stored and how it is going to be retrieved. From our experience, the use of relational databases is acceptable only if the amount of documents and speed of indexing and retrieval is not a concern. When database systems are used on large IR systems there is always a trade off between the speed of indexing and the speed of retrieval as well speed and storage efficiency.

The best approach for IR has always been a custom built inverted file structure. In the semantic role/QPLM case it is important to develop an indexing structure that can maintain the annotation information. Because it is important to allow different types of information to be indexed, we implemented a framework for information retrieval that easily incorporates different linguistic information. The framework allows fast indexing and retrieval and the implementation of different ranking strategies.

With the inclusion of relational information, the framework provides a way to retrieve documents according to a query of semantically connected words. This feature is best used when queries are formed as sentences in natural language. A simplified representation of the framework index is shown in Figure 2 for a QPLM annotated sentence. Figure 2 shows that the relation of words are represented by a common relation identifier and a relation type. The roles that each word plays in a relation is also included within the same record. The IF is optimized so that redundant information is not represented, as illustrated by the record of the word *kick* and the single document number.

QPLM representation for "*Bill kicked the ball*":

ID	Relation
11	$Who(kick) \rightarrow Bill$
12	$What(kick) \rightarrow ball$

Inverted file representation:

Term	Document	Rel. ID	Rel. Type	Role
Bill	1	11	Who	Arg
kick	1	11	Who	Pred
		12	What	Pred
ball	1	12	What	Arg

Figure 2: Simplified representation of the indexing of QPLM relations

Query	Returns documents that
$*(kick) \rightarrow *$	contain the word kick
$Who(kick) \rightarrow *$	inform that someone kicks
$Who(*) \rightarrow Bill$	inform that Bill does an action
$Who(kick) \rightarrow Bill$	inform that Bill kicks

Figure 3: QPLM Queries (asterisk symbol is used to represent a wildcard)

The framework also provides a way to include words that have not been explicitly related to other words in the text just in the same way as a standard bag-of-words (BoW) approach. This feature is important even when the text is fully semantically or syntactically parsed. Many words may not be associated with the others in a sentence because of different reasons such as errors in the parser. Therefore, even if the query presented to the retrieval component is not a proper natural language sentence or it fails to be analyzed, the system will perform as a normal BoW system.

Once the retrieval query is analyzed, it is possible to perform queries that focus on retrieving all documents where a certain relation occurs as well as all documents where a certain word plays a specific role. The example in Figure 3 demonstrates some queries and what documents or sentences they return.

A document containing the sentence "*Bill kicked the ball*" would be retrieved for all the queries in Figure 3. The framework also allows the formulation of more complex queries such as:

$$(Who(kick) \rightarrow *) \wedge (What(kick) \rightarrow ball)$$

Each token is indexed by itself (i.e not together with the related words) including the information from the relations it is part of. This is done with no overhead or redundant information being stored. This approach makes it possible to keep the standard models for document ranking. A normal calculation of Term Frequency (TF) and Inverted Document Frequency (IDF) is performed when taking the terms individually or as BoW, while only a minimal modification of TF/IDF is required when a more complex retrieval strategy is needed.

The ranking strategy is based on a vector space model. Documents and queries are represented as three different vectors: bag-of-words (BoW-V), partial relation (PR-V) and full relation (FR-V). The weights of the vector tokens are calculated using the weights of their individual tokens in the context of the vector being analyzed. In BoW-V, weights are calculated based on words; PR-V uses individual words and their relation types; FR-V uses the association of a specific word with another word. Figure 4 illustrates the contents of these vectors for the sentence "*John loves Mary, but Mary likes Brad*" when used as a query:

BoW-V:	⟨[John:1], [loves:1], [Mary:2], [likes:1], [Brad:1]⟩
PR-V:	⟨[John:ARG0:1], [loves:PRED:1], [Mary:ARG1:1], [Mary:ARG0:1], [likes:PRED:1], [Brad:ARG1:1]⟩
FR-V:	⟨[John:ARG0:loves:1], [Mary:ARG1:loves:1], [Mary:ARG0:likes:1], [Brad:ARG1:likes:1]⟩

Figure 4: Vectors used for document ranking

The tokens of the above example would have different weights if the same sentence appeared in a document with additional sentences. Because of their lower frequency, it is expected that the components of FR-V and, in a lesser extent, of PR-V to have a stronger impact on the calculation of similarity than the components of BoW-V. With this approach, for queries with relations that are not indexed, the method is equivalent to a traditional BoW approach.

4 Experiments and Evaluation

We have performed a series of experiments using the techniques described on Section 3 in order to verify the usefulness of QPLM in comparison to SRL based on PropBank. We compared both semantic annotations by using it with IR and under QA evaluation methods.

4.1 Configuration of experiments

We performed experiments using data resources from the QA track of the TREC conferences (Voorhees and Dang, 2006) and the evaluation scripts available at their TREC website of years 2004, 2005 and 2006. The retrieval experiments were carried out using only a reduced set of documents from the AQUAINT corpus because the semantic role labelers tested were not able to parse the full set, unlike QPLM which parsed all documents successfully.

The SRL tool SwiRL (Surdeanu and Turmo, 2005) has a good precision and coverage, however it is slow and quite unstable when parsing large amounts of data. We have assembled a cluster of computers in order to speed up the corpus annotation, but even when having around ten dedicated computers the estimated completion time was larger than one year. The lack of semantic annotators that can quickly evaluate large amount of data gave us the stimulus needed to use a simplified and quicker technique. We used the QPLM annotation tool which takes less than 3 weeks to fully annotate the 3GB of data from the AQUAINT corpus using a single machine.

Since we wanted to determinate how QPLM compares to SRL, particularly on the basis of its usage for IR and for QA, we performed some tests using the available amount of data annotated with semantic roles, and the same documents with QPLM. The part of the AQUAINT corpus annotated includes the first 41,116 documents, in chronological order, from the New York Times (NYT) newspaper. We used the 1,448 questions from the QA track of 2004, 2005 and 2006 from the TREC competition. Since these questions are not always self contained and in some cases (OTHER-type questions) not even a proper natural language sentence, we performed some question modification so that the entire topic text could be included. These modifications include substitution of key pronouns as well as the inclusion of the whole topic text when shorter representations were found. In some extreme cases when no substitution was possible and the question did not mention the topic, we added a phrase containing the topic at the start of the question. Some examples are presented

Topic:	Gordon Gekko
Question:	What year was the movie released?
Modification:	Regarding Gordon Gekko, what year was the movie released?
Question:	What was Gekko's profession?
Modification:	What was Gordon Gekko's profession?
Question:	Other
Modification:	Tell me more about Gordon Gekko.

Figure 5: Modifications applied to TREC questions

in Figure 5.

Using these questions as queries for our IR framework, we retrieved a set of 50 documents for every question. We analyzed the impact of the semantic annotation when used on document indices by checking the presence of the answer string in the documents returned. We also obtained a list of 50 documents using solely the BoW approach in order to compare what is the gain over standard retrieval.

4.2 Evaluation of retrieval sets

Table 1 presents the results of the retrieval set using TREC's QA track from 2004, 2005 and 2006 using the BoW, the SRL and the QPLM approaches. Because we performed the evaluation of these documents automatically, we consider a document relevant on the only basis of the presence of the required answer string. We adopted the evaluation metrics for QA documents sets proposed by Roberts and Gaizauskas (2004). We used the following metrics: $p@n$ as the precision at n documents or percentage of documents containing an answer when retrieving at most n documents; $c@n$ as the coverage at n documents or percentage of questions that can be answered using up to n documents for each question; and $r@n$ as the redundancy at n document or the average number of answers found in the first n documents per question.

As observed in Table 1, the SRL approach gives the best results for all question sets on all evaluation metrics, with the exception of $c@50$ on the 2006 question set. In most other retrieval sets the baseline performs worse than both QPLM and SRL, however for 2004 questions it performed better than QPLM on $p@50$ and $r@50$. It is interesting to observe that the QPLM results for the same year on $c@50$ are better than the BoW approach indicating that a larger amount of questions can potentially be answered by QPLM.

2004	**p@50**	**c@50**	**r@50**
BoW	5.85%	33.33%	2.92
SRL	***6.40%***	***35.33%***	***3.20***
QPLM	5.58%	34.47%	2.79
2005	**p@50**	**c@50**	**r@50**
BoW	10.03%	41.13%	5.02
SRL	***11.00%***	***43.77%***	***5.50***
QPLM	10.58%	42.08%	5.29
2006	**p@50**	**c@50**	**r@50**
BoW	7.30%	34.57%	3.65
SRL	***8.73%***	36.33%	***4.37***
QPLM	8.31%	***38.45%***	4.16

Table 1: Experimental results of index approaches on TREC questions

4.3 Experiments on QA systems

To better understand the relation between the retrieved document sets and question answering we applied the retrieval sets to four question answering systems:

- Aranea: Developed by Lin (2007), the Aranea system utilizes the redundancy from the World Wide Web using different Web Search Engines. The system relies on the text snippets to generate candidate answers. It applies filtering techniques based on intuitive rules, as well as the expected answer classes with named-entities recognition defined by regular expressions and a fixed list for some special cases.

- OpenEphyra: Developed by Schlaefer et al. (2007), the OpenEphyra framework attempts to be a test bench for question answering techniques. The system approaches QA in a fairly standard way. Using a three-stage QA architecture (Question Analysis, Information Retrieval, Answer Extraction), it performed reasonably well at the QA Track at TREC 2007 by using Web Search engines on its IR stage and mapping the answers back into the TREC corpus.

- MetaQA System: Similar to the Aranea QA system, MetaQA (Pizzato and Molla, 2005) makes heavy use of redundancy and the information provided by Web Search Engines. However it goes a step further by combining different classes of Web Search engines (including Web Question Answering Systems) and assigning different confidence scores to each of the classes.

- AnswerFinder: Developed by Mollá and Van Zaanen (2006), the AnswerFinder QA system unique feature is the use of QA graph rules learned automatically from a small training corpus. These graph rules are based on the maximum common subgraph between the deep syntactic representation of a question and a candidate answer sentence. The graphs were derived from the output of the Connexor dependency-based parser.

For most of these systems some modifications of the standard system configuration were required. All the systems used, with the exception of AnswerFinder, make heavy use of web search engines and the redundancy obtained to find their answers. For our experiments we had to turn the Web search off, causing a significant drop in performance when compared to the reported results in the literature. Because AnswerFinder's IR component is performed offline, the integration is seamless and only required providing the system with a list of documents in the same format as TREC distributes the ranked list of files per topic. The OpenEphyra framework is well designed and implemented, however the interaction between its components still depended on the overall system architecture, which makes the implementation of new modules for the system quite difficult.

With the exception of AnswerFinder, all the QA systems received a retrieval set as a collection of snippets. This was based on the fact that these systems are based on Web Retrieval and they expect to receive documents in this format. We extracted for every document the 255 character window where more question words (non-stopwords) were found. The implementation of different ranking strategies for passage retrieval such as those described by Tellex et al. (2003) could improve the results for individual QA systems. However, a preliminary evaluation of the passage retrieval have shown us that the 255 character window with the current snippet construction method was enough to achieve near optimal performance on the document set used.

The results obtained by the QA systems were processed using the answer regular expressions distributed by TREC. The numbers described in this study show the factoid score for correct answers. We have not used the exact answer because it required some cleaning of the answer log files and some modification of some QA systems.

	2004	**2005**	**2006**
BoW	5.00%	2.30%	2.10%
SRL	***6.10%***	***3.50%***	2.70%
QPLM	5.00%	2.50%	***3.50%***

Table 2: Factoid results for $C@1$ on the Aranea system

	2004	**2005**	**2006**
BoW	2.50%	5.10%	3.00%
SRL	***3.30%***	***7.00%***	***4.40%***
QPLM	2.80%	6.20%	4.20%

Table 3: Factoid results for $C@1$ on the OpenEphyra system

Therefore, the results shown on Tables 2, 3 and 5 are product of the same retrieval set and result of the same evaluation procedure. Results of the MetaQA system at Table 4 are presented as coverage at answer 10 ($C@10$) since this system has a non standard approach for QA that is invalidated by the methodology of this test. The results in the other tables could be understood as either precision or coverage at answer 1, we will refer to them as $C@1$.

We observed that the results from the QA system are consistent with the findings from the results of the retrieval system. The Aranea QA system results on Table 2 show an average improvement for the SRL approach. QPLM has similar performance to BoW for 2004 and 2005 questions but outperforms both techniques on 2006 questions.

The results shown by OpenEphyra in Table 3 also demonstrate that semantic annotation can help question answering when used in the IR stages of a QA system. The best results were observed when SRL was applied. QPLM followed SRL and outperformed BoW on three tests. It is important to point out that results for the retrieval set alone in Table 1 showed BoW outperforming QPLM for 2004 questions on both redundancy and precision metrics. This might be an indication that OpenEphyra answer extraction modules are more precise than the other QA systems and do not heavily rely on redundancy as do the Aranea and the MetaQA systems.

Because of the high dependency on Web sources, the MetaQA system performed rather poorly. As explained earlier, the results were measured using $C@10$ instead of $C@1$. The reason for this is that the MetaQA system is meant to be an aggregator of information sources and its ranking

	2004	2005	2006
BoW	0.87%	3.31%	1.24%
SRL	***2.61%***	***3.87%***	***1.99%***
QPLM	0.43%	3.31%	1.24%

Table 4: Factoid results for $C@10$ on the MetaQA system

	2004	2005	2006
BoW	1.10%	2.50%	1.20%
SRL	***1.80%***	2.60%	***2.20%***
QPLM	***1.80%***	***2.70%***	2.00%

Table 5: Factoid results for $C@1$ on the AnswerFinder system

mechanisms only work when sufficient evidence is given for certain entities. Not only was the system not designed for the single-source setup, but it was not designed to provide a single answer. Nevertheless, even with the non-conformity of the system, it appears to support that semantic markup can enhance the IR results for QA. Not surprisingly the extra redundancy presented in the 2004 BoW retrieval contributed to better results in this redundancy based QA system.

Results in Table 5 show that AnswerFinder correctly answered only a few questions for the given question set. On the other hand, it provided some consistent results such that the improvements were due to additional correct answers and not to a larger but different set of correct answers. The AnswerFinder QA system showed a similar performance for both semantic-based strategies and both outperformed the BoW strategy.

In this section we have shown an evaluation of different retrieval sets of documents using four distinct QA systems. We have observed that semantic strategies not only assist the retrieval of better documents, but also help in finding answers for questions when used with QA systems.

5 Concluding Remarks

In this work we propose the use of semantic relation in QA. We also present QPLM as an alternative to SRL. QPLM is a simpler approach to semantic annotation based on relations between pairs of words, which gives a large advantage in speed performance over SRL. We show some comparison of retrieval sets using the questions from the QA track of TREC and conclude that SRL and QPLM improve the quality of the retrieval set over a standard BoW approach. From these results we also observe that QPLM performance does not fall much behind SRL.

We performed an evaluation using four QA systems. These systems are conceptually different which gives a broad perspective of the obtained results. The results once again show the effectiveness of semantic annotation. Over QA, SRL has performed better than the other techniques, but was closely followed by QPLM. The results obtained here suggest that QPLM is a cheaper and effective method of semantic annotation that can help in tuning the search component of a QA system to find the correct answers for a question.

The results presented in this work for all QA systems are much lower than those reported in the literature. This is an undesirable but expected problem that occurred not only because of the modifications carried on the QA systems but mainly because of the reduced number of documents used for this evaluation. We are looking into more efficient alternatives for performing the SRL annotation of the AQUAINT corpus.

Only recently we have been able to test Koomen et al. (2005) SRL tool. This SRL tool is the top ranking SRL tool at the CoNLL-2005 Shared Task Evaluation and it seems to be much faster than SwiRL. Preliminary tests suggest that it is able to perform the annotation of AQUAINT in almost one full year using a single computer; however, this tool, like SwiRL, is not very stable, crashing several times during the experiments. As further work, we plan to employ several computers and attempt to parse the whole AQUAINT corpus with this tool.

It is important to point out that although the tool of Koomen et al. seems much faster than SwiRL, QPLM still outperforms both of them on speed by large. QPLM represents word relations that are built using rules from syntactic and NE information. This simpler representation, combined with a smaller number of supporting NLP tools, allow QPLM to be faster than current SRL tools. We plan to carry out further work on the QPLM tool to increase its performance on both speed and accuracy. QPLM's precision and recall figures are going to be improved by using a hand annotated corpus. QPLM's speed suggest that it can be currently used on IR tools as a pre-processing engine. It is understandable that any delay in the IR phases is undesirable when dealing with large amount of data, therefore optimizing the speed of QPLM is one of our priorities.

Acknowledgement

This work was supported by an iMURS scholarship from Macquarie University and the CSIRO.

References

Baker, Collin F., Charles J. Fillmore, and John B. Lowe. 1998. The berkeley framenet project. In *Proceedings of the 17th international conference on Computational linguistics*, pages 86–90, Morristown, NJ, USA. Association for Computational Linguistics.

Barzilay, Regina, Kathleen R. McKeown, and Michael Elhadad. 1999. Information fusion in the context of multi-document summarization. In *Proceedings of the 37th annual meeting of the Association for Computational Linguistics on Computational Linguistics*, pages 550–557, Morristown, NJ, USA. Association for Computational Linguistics.

Graff, David. 2002. The AQUAINT corpus of english news text. CDROM. ISBN: 1-58563-240-6.

Kaisser, Michael and Bonnie Webber. 2007. Question answering based on semantic roles. In *Proceedings of the ACL 2007 Workshop on Deep Linguistic Processing,*, page 4148, Prague, Czech Republic, June. c2007 Association for Computational Linguistics.

Kogan, Y., N. Collier, S. Pakhomov, and M. Krauthammer. 2005. Towards semantic role labeling & ie in the medical literature. In *American Medical Informatics Association Annual Symposium.*, Washington, DC.

Koomen, P., V. Punyakanok, D. Roth, and W. Yih. 2005. Generalized inference with multiple semantic role labeling systems (shared task paper). In Dagan, Ido and Dan Gildea, editors, *Proc. of the Annual Conference on Computational Natural Language Learning (CoNLL)*, pages 181–184.

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

Litkowski, K. 1999. Question-answering using semantic relation triples. In *In Proceedings of the 8th Text Retrieval Conference (TREC-8*, pages 349–356.

Molla, Diego and Menno van Zaanen. 2006. Answerfinder at TREC 2005. In *The Fourteenth Text REtrieval Conference (TREC 2005)*, Gaithersburg, Maryland. National Institute of Standards and Technology.

Narayanan, Srini and Sanda Harabagiu. 2004. Question answering based on semantic structures. In *COLING '04: Proceedings of the 20th international conference on Computational Linguistics*, page 693, Morristown, NJ, USA. Association for Computational Linguistics.

Palmer, Martha, Daniel Gildea, and Paul Kingsbury. 2005. The proposition bank: An annotated corpus of semantic roles. *Computational Linguist.*, 31(1):71–106.

Pizzato, Luiz Augusto and Diego Molla. 2005. Extracting exact answers using a meta question answering system. In *Proceedings of the Australasian Language Technology Workshop 2005 (ALTA-2005).*, The University of Sydney, Australia, December.

Pizzato, Luiz Augusto and Diego Mollá. 2007. Question prediction language model. In *Proceedings of the Australasian Language Technology Workshop 2007*, pages 92–99, Melbourne, December.

Quantz, Joachim and Birte Schmitz. 1994. Knowledge-based disambiguation for machine translation. *Minds and Machines*, 4(1):39–57, February.

Roberts, Ian and Robert J. Gaizauskas. 2004. Evaluating passage retrieval approaches for question answering. In McDonald, Sharon and John Tait, editors, *Advances in Information Retrieval, 26th European Conference on IR Research, ECIR 2004, Sunderland, UK, April 5-7, 2004, Proceedings*, volume 2997 of *Lecture Notes in Computer Science*, pages 72–84. Springer.

Schlaefer, N., P. Gieselmann, and G. Sautter. 2007. The ephyra qa system at trec 2006 the Ephyra QA system at TREC 2006. In *The Fifteenth Text REtrieval Conference (TREC 2006)*.

Shen, Dan and Mirella Lapata. 2007. Using semantic roles to improve question answering. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 12–21, Prague, June 2007. Association for Computational Linguistics.

Sun, R. X., J. J. Jiang, Y. F. Tan, H. Cui, T. S. Chua, and M. Y. Kan. 2005. Using syntactic and semantic relation analysis in question answering. In *Proceedings of the TREC*.

Surdeanu, Mihai and Jordi Turmo. 2005. Semantic role labeling using complete syntactic analysis. In *Proceedings of CoNLL 2005 Shared Task*, June.

Tellex, Stefanie, Boris Katz, Jimmy Lin, Aaron Fernandes, and Gregory Marton. 2003. Quantitative evaluation of passage retrieval algorithms for question answering. In *SIGIR '03: Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval*, pages 41–47, New York, NY, USA. ACM Press.

Voorhees, Ellen M. and Hoa Trang Dang. 2006. Overview of the TREC 2005 question answering track. In *Text REtrieval Conference*.

Author Index