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.

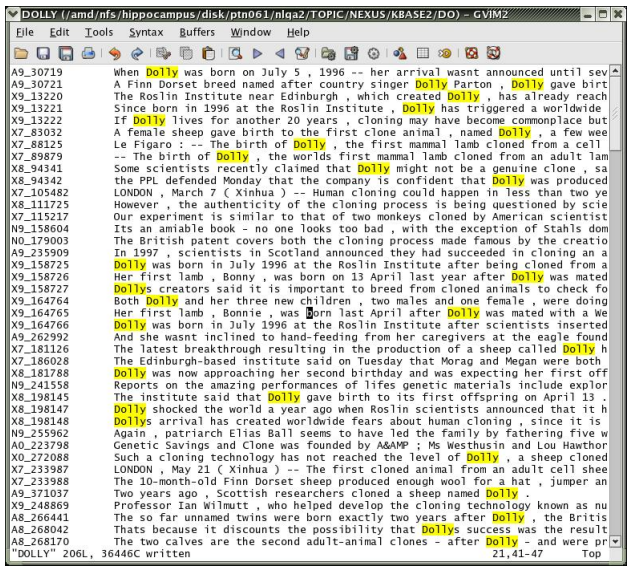

Figure 1: An Example Topic Document: Dolly the sheep

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.

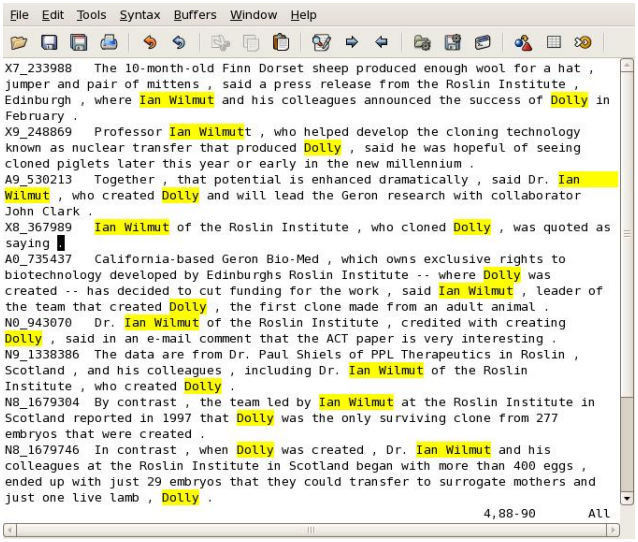

Figure 2: A Bi-Topic Document: (Dolly, Ian Wilmut)

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.