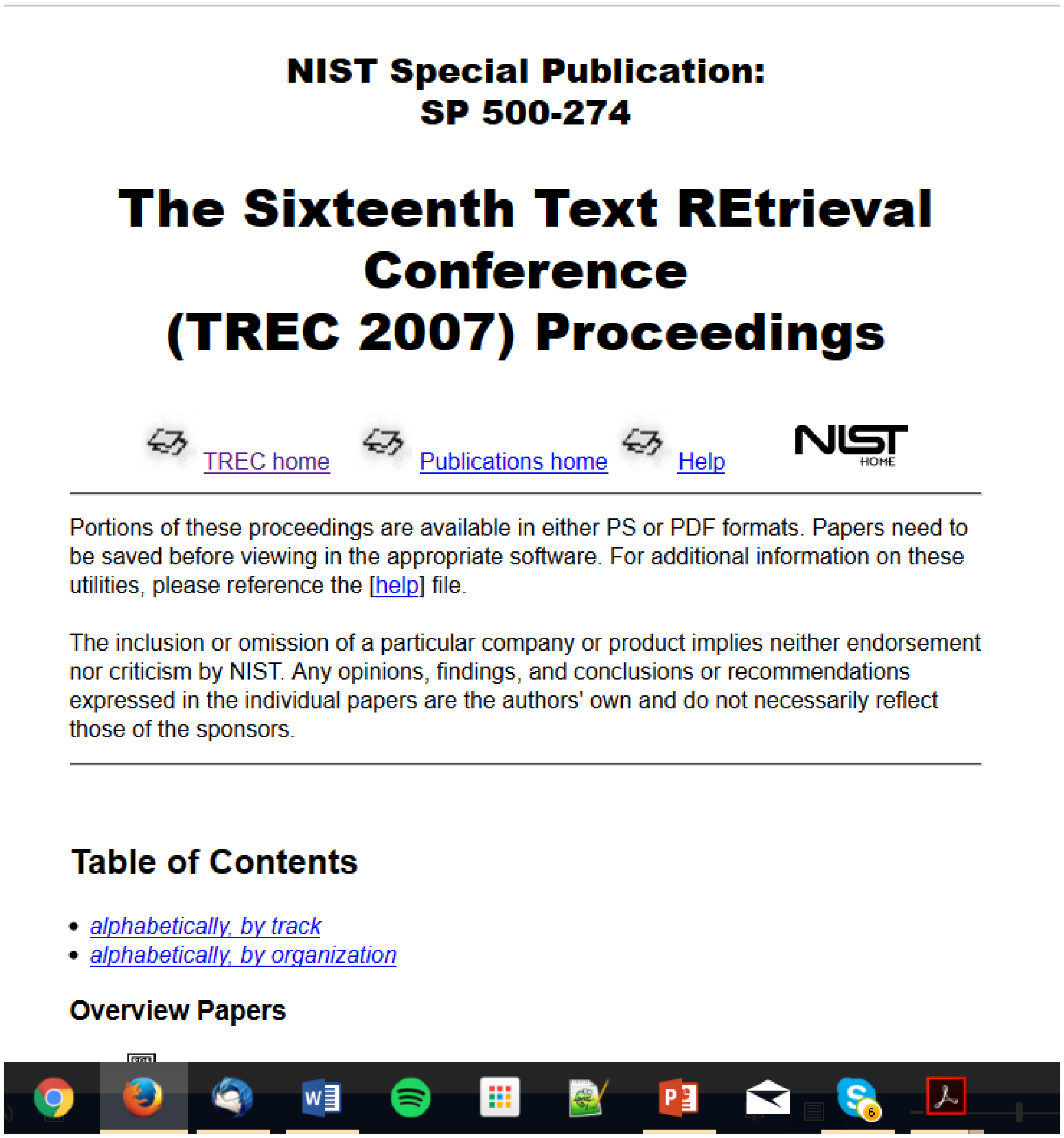

NIST Special Publication:
SP 500-274

The Sixteenth Text REtrieval Conference (TREC 2007) Proceedings

TREC home Publications home Help

Portions of these proceedings are available in either PS or PDF formats. Papers need to be saved before viewing in the appropriate software. For additional information on these utilities, please reference the [help] file.

Table of Contents

Overview Papers

Concordia University at the TREC 2007 QA track

Majid Razmara, Andrew Fee and Leila Kosseim
CLaC Laboratory
Department of Computer Science and Software Engineering
Concordia University
1455 de Maisonneuve Blvd. West
Montral, Québec, Canada, H3G 1M8
`m_razma; a_fee; kosseim @cse.concordia.ca`

Abstract

In this paper, we describe the system we used for the TREC-2007 Question Answering Track. For *factoid* questions our redundancy-based approach using a modified version of ARANEA was enhanced further. Our *list* question answerer uses a clustering method to group the candidate answers that co-occur together. It also uses the target and question keywords as *spies* to pinpoint the right cluster of candidates. To answer *other* types of questions, our system extracts from Wikipedia articles a list of *interest-marking* terms and uses them to extract and score sentences from the AQUAINT-2 and BLOG document collections using various interest-marking triggers.

1 Introduction

This year, we continued improving our QASCU system on factoid and *other* questions and we developed a new subsystem to deal with list questions since we had spent very little time on it last year. As last year's system [1] received relatively good results in general, this year we decided to focus more on list questions, where we did poorly, and improve the factoid and *other* modules of the system. Our system for factoid questions, used to exploit two main approaches: a redundancy-based QA system working on the Web (a modified version of ARANEA [2, 3, 4]) and a linguistic-based system working on AQUAINT-2 only. Since the linguistic-based approach did not improve the results significantly, we only concentrated on the redundancy-based approach that seemed more promising. We spent about 1 person-month improving ARANEA and modifying it for this year's TREC.

The list question answerer is built on top of our factoid question answerer to receive a list of candidate answers and filter appropriate candidates. It also extracts a list of possible answers and adds them to the provided candidate list. Two person-months were spent on the list question answerer.

We have not significantly changed our *other* question answerer since last year; merely, small improvements in query generation and other parts were made. For the *other* questions, we used terms extracted from Wikipedia and projected them on the AQUAINT-2 collection. Sentences containing these Wikipedia terms are then ranked using various interest-marking triggers.

In the following sections, we describe our question answering system. In section 2, we describe how we answer factoid questions; in section 3, we describe the list questions module and in section 4, the *other* questions module. Finally, we present our results in section 5.

2 Answering Factoid Questions

This year two person-months were spent improving last year's system and modifying it for the TREC-16 task specifications. The Question Answering System of Concordia University (QASCU) uses a redundancy-based approach working on the World Wide Web to answer factoid questions. It is a modified version of Jimmy Lin's ARANEA question answering system. ARANEA is a web-based rule and statistical QA system. A simplistic overview of the system is that it generates queries and question reformulations, gets snippets from Google and Teoma, generates n-grams as possible answers, collates the n-grams based on their similarity and does filtering based on the expected type of answer. Detailed descriptions of ARANEA can be found in [2, 3, 4].

While the overall architecture of this subsystem was not significantly changed, many modifications were made to the existing modules to contribute to an improvement in the overall system accuracy. A new component was also introduced to tackle the problem of selecting the most suitable supporting document for each answer.

Apart from these modifications, many non-performance-enhancing changes were made to accommodate the new corpora introduced this year. For the TREC-16 competition, the corpora consisted of the AQUAINT-2 text research collection and the BLOG web research collection. We did not purchase the BLOG collection, but NIST provided the text of the top 50 documents per target (as retrieved by the PRISE search engine when using the target as the query). Following are the main modifications made to last year's QASCU system.

2.1 Predicting Answer Types

The factoid question answerer subsystem generates expected answer types through a simple word match and ordering algorithm. Possible expected answer types include: DATE, PERSON, LOCATION, ORGANIZATION, WEBSITE, NUMBER (including the unit, if applicable), and OTHER. The generation of an expected answer type is an important early step in the question answering process. Questions of different types are handled very differently by the system. Candidate answers that do not match the semantic type expected by the question are filtered out. By analyzing the results of QASCU in the TREC-15 competition, it was found that correctly classified questions had double the probability of being answered correctly when compared to incorrectly classified questions.

The type classification was improved this year by adding about 50 new words to the word-match dictionary and modifying the algorithm. By comparing the results of the typing modules with a manual type classification of the questions, a 6% improvement was observed (from an accuracy of 87% using the old module, to 93% with the new module) over the same set of 963 questions.

2.2 5-gram Generation

One key step in the pipeline is the generation of all possible n-grams from the text fragments generated by the request execution module. QASCU 2006 had an upper limit of 4-grams. This caused some questions to be incorrectly answered since the required answer contained five words. Facilitating the generation of 5-gram candidates solved this problem and did not have a detrimental effect on other questions with shorter answers.

2.3 Website Questions

Every year, there are a few questions that ask for a company or organization's web address. It was decided that these questions can be better answered if they are not treated like typical questions. A new Google query was specifically formed for these questions. The target of the question (i.e. the organization's name) was paired with the word "website" and this was used to search Google. The domain name of the first returned URL is taken as the candidate answer.

2.4 Question-Specific Filtering methods

Depending on the expected answer type of a question, the list of candidate answers is filtered in order to decrease the number of potential answers. The goal is to improve the accuracy of the system by using simple heuristics to eliminate obviously wrong answers and also answers of an invalid format. A lot of time was spent on expanding these existing rule sets by examining the TREC data sets from 2004, 2005, and 2006. Some highlights include:

Number questions Typically number questions ask about a specific quantity (e.g *What was the distance of the 1999 Chicago Marathon?* or *What is the company's annual revenue?*). Candidate answers to number questions can be filtered based on the expected units of the number. Several new sets of units were added to the system including: length, temperature, acceleration, velocity, time, mass, power, money, and age.

Closed-class questions A closed-class question is one with a set of possible answers that can be exhaustively enumerated. Several new gazetteer lists were added to the system to improve various closed-class questions (e.g. U.S. television networks; professional baseball, hockey, football and basketball teams; major cities from around the world; U.S. state capitals).

Often it is not feasible to completely enumerate all possible answers in a closed-class domain. In a case like this we use candidate answer promotion as opposed to filtering. The idea is to boost the rank of certain candidate answers that appear in the closed-class set (instead of filtering out those that do not). That way, if a correct candidate answer does not appear in the closed-class set for some reason, it's not completely removed and there is a chance to re-rank the answers later in the pipeline.

Acronym questions Another common type of question is one that asks what a certain acronym stands for. One rule that ended up working quite well was to keep only candidate answers that consist of capitalized words; each starting with a letter matching the acronym pattern. We skip over any internal prepositions since these words are typically not included in an acronym. Conversely, there are questions that ask for the acronym of an organization's name. One rule that worked well was to eliminate candidate answers that were not entirely comprised of uppercase letters, and answers that did not at least begin with the same letter as the long-form name.

2.5 Answer Projection

The use of a better answer projection step was explored this year. This process involves finding the best supporting document from the AQUAINT-2 or BLOG corpus to be paired with the web-derived answer. Initially, documents are retrieved from the corpora using the Lucene search engine. For each document returned we store the name, the Lucene score, and a short raw text snippet (called the "page summary"). Later in the pipeline, we gather a list of supporting documents for the top-ranked candidate answer by going through all the page summaries and selecting those documents in which the answer string (or an inexact version of it) appears at least once.

The previous version of QASCU used an approach to finding the best supporting document by simply selecting the supporting document with the highest Lucene score. However, this method is not ideal since the Lucene score considers the TF-IDF of each term in the search query, but does not take into account the frequency of the terms in the answer string.

Many different methods of answer projection were explored this year. The techniques were all variations on a ranking system which involved the frequency of both the answer string and the question target in each document in the supporting document list. We began with the generation of an inexact version of both the answer string and the question target. The inexact version is a subset of the string (generated differently depending on the expected answer type). The full text of each supporting document was then retrieved. The document with the highest combined frequency of the answer string + the target, multiplied by the Lucene score, was selected as the best supporting document.

The new answer projection module performed adequately. However, its performance in selecting the correct supporting document only matched that of the previous year's approach (for the 2004 - 2006 TREC data). The system selects the correct document for approximately 60% of the correctly-answered.

3 Answering List Questions

Contrary to last year, this year we focused on list questions. The goal was to come up with a new approach to select the candidate answers, having an initial list of potential answers, with exploiting merely statistical techniques. We hypothesized that different instances of an answer to a list question have a special relation one another. Beside the fact that they are all of the same entity class (e.g. country names, people, book titles, ...), they co-occur within the sentences of the documents related to the target and the question. In addition, the instances of answers also tend to co-occur with the target and question keywords.

Figure 1 shows a few sample snippets from AQUAINT-2 and the web related to question 232.6 *Which airlines use Dulles?* (Target: *Dulles Airport*). As the figure shows, the snippets contain a few instances of answers along with the target and question keywords.

We propose a novel approach based on the hypothesis that different instances of an answer to a list question should appear in close proximity of one another and also of the target and

Target 232: "Dulles Airport" *Question 232.6: "Which airlines use Dulles?"*

Source	Snippet Containing Answers
LTW 20050712.0032	**United**, which operates a hub at Dulles ... **Delta**, Northwest, **American**, **British Airways** and **KLM** share four screening machines in the basement.
TTW 20060102.0106	**Flyi** suffered from rising jet fuel costs and the aggressive response of competitors, led by **United** and **US Airways**. They matched **Independence**'s fares, ... from Dulles Airport to cities such as Newark.
NYT 20050426.0151	**Delta** recently added more than 80 flights from its Atlanta hub and capped its business fares, while money-losing **Independence Air**, based at Washington's Dulles Airport, is making ....
WIKIPEDIA	At its peak of 600 flights daily, **Independence**, combined with service from **JetBlue** and AirTran, briefly made Dulles the largest low-cost hub in the United States.
NEW YORK TIMES	**Continental Airlines** sued **United Airlines** and the committee that oversees operations at Washington Dulles International Airport yesterday, ...

Figure 1: Answers tend to co-occur with one another and with the target and question keywords (acceptable answers are shown in bold face and the target and question keywords are underlined)

the question keywords. These distinct answers may occur in most documents related to the target. However, co-occurrence can have different scopes: co-occurrence within a document, a paragraph, a sentence or a window of words. Our system in TREC 2007 only considers co-occurrence at the sentence level. Although worth trying, the other possibilities were not implemented due to time constraints. Of course, they will be considered in future work. The overall architecture of our *list* answerer subsystem is illustrated in Figure 2.

3.1 Candidate Answers Extraction

The first step to creating a list of candidate answers is answer type recognition. Each question is associated to one of the nine entity classes: PERSON, COUNTRY, ORGANIZATION, JOB, MOVIE, NATIONALITY, CITY, STATE, OTHER. This is done by using lexical and syntagmatic patterns. Once the type of answer is predicted, a number of documents are retrieved from AQUAINT-2 and the web using a query generated from the target and the question. These documents constitute a collection from which candidate terms are extracted. All terms that conform to the answer type are extracted from this collection. Depending on the answer type, the candidate terms are extracted using an NE tagger (in case of PERSON, ORGANIZATION and JOB), using a gazeteer (in case of COUNTRY, NATIONALITY, STATE and partially CITY) and finally extracting all capitalized terms and terms in quotations and validating them using web frequency (in case of MOVIE and OTHER).

The candidate answers extracted in this module are added to those extracted using our factoid question answerer subsystem and together constitute our final candidate answers.

3.2 Relation Extraction

The relation between two candidate terms is a normalized value denoting how often they co-occur within documents about the target. For this purpose, using the query generated in the previous section, a list of relevant documents from AQUAINT-2 and the web are retrieved. This constitutes the domain collection from which sentences will be extracted to compute

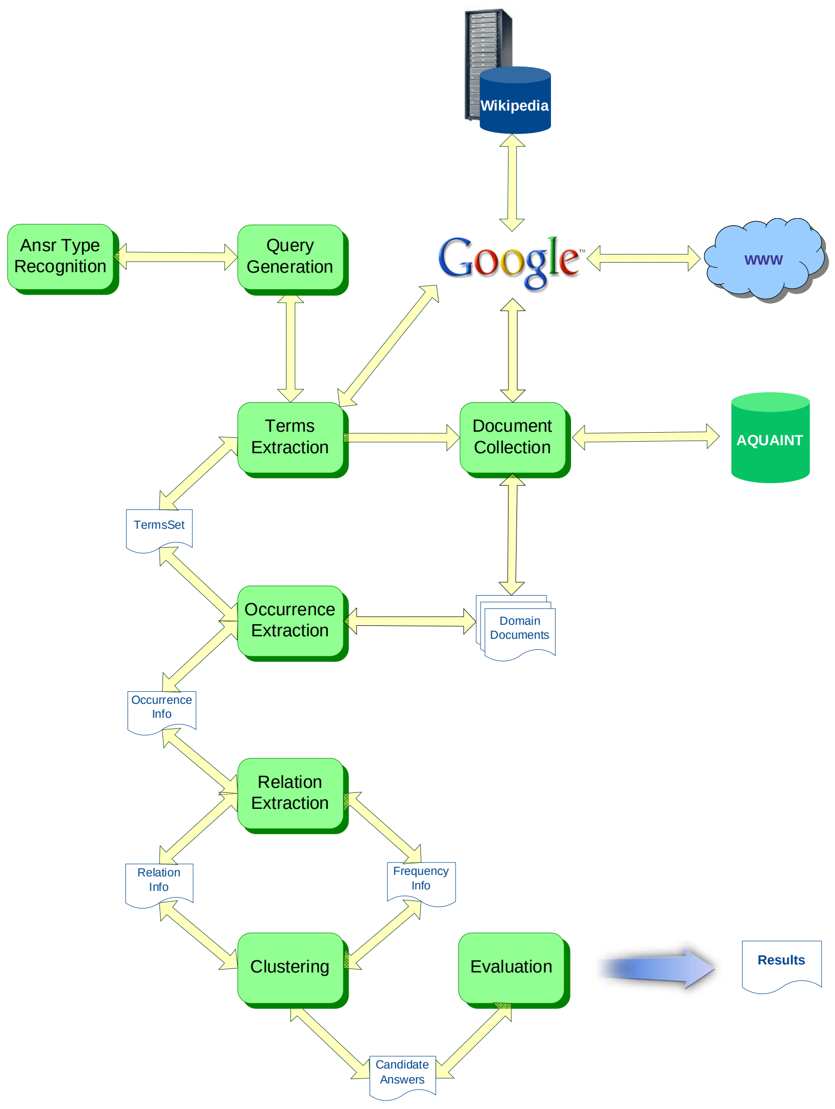

Figure 2: List Answering Subsystem Architecture

co-occurrence information. Once all the data regarding term occurrences and co-occurrences is collected, the relation between each pair of term is computed using the chi-square (χ^2) test.

$$\chi^2 = \frac{N\,(O_{11}O_{22} - O_{11}O_{21})^2}{(O_{11}+O_{12})(O_{11}+O_{21})(O_{12}+O_{22})(O_{21}+O_{22})}$$

Where O_{11} refers to the number of sentences $term_i$ and $term_j$ appeared together and O_{12} refers to the number of sentences in which $term_i$ appeared but $term_j$ did not appear. Similarly O_{21} is the number of sentences containing $term_j$ but not $term_i$ and O_{22} is the number of sentences containing neither $term_i$ nor $term_j$. N denotes the total number of sentences in the domain collection.

3.3 Clustering

Once a table of candidate terms and the relations among them is available, a clustering method tries to create a subset of the candidate instances which have a higher probability to be the correct answer. We use a Hierarchical Agglomerative Average-Linkage (HAAL) clustering. The algorithm is as follows:

1. Put each candidate term t_i in a separate cluster C_i.
2. Compute the relation between each two clusters. In average-linkage clustering, the relation between two clusters C_i and C_j is the average of all relations between terms t_m in C_i and terms t_n in C_j.

$$relation(C_i, C_j) = \frac{1}{|C_i| \times |C_j|} \sum_{t_m \in C_i} \sum_{t_n \in C_j} relation(t_m, t_n)$$

3. Merge two clusters which have the highest relation between them.
4. Goto step 2 until the highest relation is below a certain threshold.

After the clustering is finished, the main concern is how to select the right cluster. For this purpose, before clustering, we add the target and question keywords to our candidate terms to be clustered. Their responsibility is to *spy* on candidate terms. These spies are treated exactly like candidate terms; hence their co-occurrences with candidate terms and also other spies are extracted, their relations are evaluated and finally they are clustered along with candidate terms. When clustering is finished, these *spies* are used to pinpoint the cluster with the highest probability of being the correct cluster. This is according to our hypothesis that the answers to a list question tend to co-occur with one another and with the target and question keywords as well.

A second approach to narrow down the primary candidate terms was exploited. This approach simply selects the terms whose overall relation to other terms is the highest. We call this approach CRV as it uses the Cumulative Relation Value to compare terms one another. Surprisingly, this approach also worked out (see section 5).

3.4 Answer Projection

To support the final answers with documents from AQUAINT-2 and/or BLOG, a simple method is used. The corpora are searched using the query generated before and the final candidate term and simply the first document is returned as the supporting document. Since we did not have access to the whole BLOG corpus, even though the top 50 documents were available, we focus more on the AQUAINT-2 corpus to return the supporting documents. Several different approaches were examined that all caused a lower score than this method.

4 Answering Other Questions

This year, we didn't invest too much effort on *other* questions since our results for TREC 2006 were quite satisfactory. Only two person-days were spent to improve different components of the system. A brief summary of *other* section is provided. For more details about how *other* questions are answered, please refer to [5].

Fundamentally, we hypothesized that interesting nuggets can be extracted using two types of interest markers:

Target-specific interest markers: terms that are important within the documents related to the target. For example, *sinking* is an interesting term in the target "Titanic" or *assassination* contains a valuable data about "Kennedy".

Universal interest markers: terms that are important regardless of the target. For example, in the sentence "first man on the moon", *first* conveys an interesting concept or in the sentence "15000 people died in the yesterday's earthquake disaster", *15000* contains a unique meaning.

To identify target-specific interest marking terms, we used the Wikipedia[1] online dictionary. The first stage to answering *other* questions is to find the proper Wikipedia article. We use the Google API to search in the Wikipedia domain using the target as query. The first Wikipedia article that satisfies the query is taken. We extract named entities as interesting terms for each target, and we search AQUAINT-2 for the N most relevant documents. For this purpose, the title of the Wikipage is added to the query.

Within the documents chosen as the domain, the frequency of each interest marking term is then computed. For each term, we compute a weight as the logarithm of its frequency.

$$Weight(T_i) = Log(Frequency(T_i))$$

All sentences from the domain documents are then scored according to how many target-specific interesting terms it contains. This is computed as the sum of the weight of the interesting terms it contains.

$$Score(S_i) = \sum_{T_j \in S_i} Weight(T_j)$$

Then similar sentences that either are almost equivalent to one other at the string level or share similar words but not the same syntax are dropped.

Once the sentences are ranked based on target-specific interesting terms, we boost the score of sentences that contain terms that generally mark interesting information regardless of the topic. Such markers were determined empirically by analyzing the previous TREC data. These markers consists of superlatives, numeral and target-type specific keywords. This last type of marker is essentially a list of terms that do not fit any specific grammatical category, but just happen to be more frequent in interesting nuggets. Finally, the top N sentences making up 7000 non-white-space characters are returned as our nuggets.

5 Results

Table 1 shows the official evaluation results of our runs along with the median and best score of all systems. Following is the description of all the runs for each question type.

[1] `http://en.wikipedia.org`

	QASCU1	QASCU2	QASCU3	**median**	**best**
Factoid	0.256	0.242	0.213	0.131	0.706
Incorrect	227	231	239		
Unsupported	25	26	25		
Inexact	13	29	15		
Locally correct	3	1	2		
Globally correct	92	87	79		
list	0.128	0.134	0.145	0.085	0.479
Other	0.275	0.281	0.278	0.118	0.329
Average per-series	0.222	0.221	0.214	0.108	0.484

Table 1: Official results of the 3 runs.

5.1 Factoid Runs

In the QASCU1 run, only AQUAINT-2 and the web are used for information retrieval. We use the top 50 documents retrieved by PRISE for candidates re-ranking. Candidate answers are re-ranked using frequency of the terms in the top 50 documents returned by PRISE. We use only AQUAINT-2 to select the supporting documents for the top-ranked answers.

In the QASCU2 run, information retrieval is the same as in QASCU1. However, candidates re-ranking is based on frequency of the words in the whole AQUAINT-2 corpus. Answer projection is done on AQUAINT-2 and the top 50 BLOG corpus.

QASCU3 uses AQUAINT-2, the top 50 BLOG documents and the web for information retrieval. Candidates re-ranking is based on frequency of the words in the top 50 PRISE documents. Document supporting source is the same as QASCU2.

5.2 List Runs

QASCU1 uses the HAAL clustering algorithm to cluster the primary candidate answers. For answer projection, only the AQUAINT-2 documents are taken into account (not the BLOG corpus). In QASCU2, CRV method is used to select the final candidate terms and again only AQUAINT-2 is used as the source of supporting documents. However, QASCU3 uses both AQUAINT-2 and BLOG for answer projection. In this run, the candidates of our factoid answerer system is also added to the candidate list and the CRV method is used to choose the final answers.

5.3 Other Runs

We only submitted two runs for *other* questions. QASCU1 extracts the candidate terms from Wikipedia while QASCU2 uses Wikipedia and also two documents from AQUAINT-2 as the source of term extraction. QASCU3 is the same as QASCU1 although its F-score is slightly different. This is due to some minor changes in the document list returned by Google in different times.

6 Conclusion

In this paper, we described our approach to answering different types of questions in QA track of TREC. The factoid answerer subsystem is based on a modified version of ARANEA. The list answerer narrows down the list of candidate answers in order to get a higher precision. Our system approach for answering *other* questions is based on terms found in Wikipedia entries and ranking of nuggets is done through the use of target-specific and target-independent interest markers.

Although our system outperforms the median scores in all three types of questions, it can be improved in its algorithms, methods and components.

References

[1] L. Kosseim, A. Beaudoin, A. K. Lamjiri, and M. Razmara. Concordia University at the TREC-QA Track. In *n Proceedings of the 15th Text Retrieval Conference (TREC-15)*, Gaithersburg, USA, November 2006.

[2] Boris Katz, Jimmy Lin, Daniel Loreto, Wesley Hildebrandt, Matthew Bilotti, Sue Felshin, Aaron Fernandes, Gregory Marton, and Federico Mora. Integrating web-based and corpus-based techniques for question answering. In *Proceedings of the 12th Text Retrieval Conference (TREC-12)*, 2003.

[3] Boris Katz and Jimmy Lin. Question answering from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the twelfth international conference on Information and knowledge management (CIKM)*, 2003.

[4] Boris Katz, Gregory Marton, Jimmy Lin, Aaron Fernandes, and Stefanie Tellex. Extracting answers from the web using knowledge annotation and knowledge mining techniques. In *Proceedings of the 11th Text Retrieval Conference (TREC-11)*, 2002.

[5] M. Razmara and L. Kosseim. A little known fact is . . . Answering Other questions using interest-markers. In *Proceedings of the 8th International Conference on Intelligent Text Processing and Computational Linguistics (CICLing-2007), pages 518-529*, Mexico, 2007.