Concordia University at the TAC-2008 QA Track

Majid Razmara[*] and Leila Kosseim
CLaC Laboratory
Department of Computer Science and Software Engineering
Concordia University
1455 de Maisonneuve Blvd. West
Montréal, Québec, Canada, H3G 1M8
mra44@sfu.ca; kosseim@cse.concordia.ca

Abstract

In this paper, we describe the system we used for the TAC-2008 Question Answering track. Our *Rigid* question answerer uses a classification method to group candidate answers into *relevant* and *irrelevant* classes based on the co-occurrence information of each pair of candidate answers. To answer *Squishy* 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 BLOG06 document collections using various interest-marking triggers.

1 Introduction

This year, NIST introduced a novelty to the TREC QA track named TAC QA track. It brought question-answering researchers two new challenging types of questions; *Rigid* as a successor to *List* questions with addition of opinion information and *Squishy* as a successor to *Other* questions with specification of the interest of the question.

Since we achieved relatively good results at TREC-2007 with our *List* and *Other* question answering subsystems [1], we decided to exploit our subsystems from last year and to add on top of them new features to deal with answering *Rigid* and *Squishy* questions specifically. We had planned to use sentiment information to filter out candidate answers that do not come from sentiment-bearing sentences. We started to use terms from the General Inquirer[1] and the results of the Opinion Retrieval Task of the TREC 2006 Blog Track [2] to learn to classify sentences as opinion-bearing or not. By using last year's subsystems for *List* and *Other* questions, we would produce a list of candidate answers, and then filter out those candidates that do not come from an opinion-bearing sentence. Unfortunately, due to time constraints and technical difficulties, we were unable to submit this run, and therefore only relied on last year's system with very minor modifications.

In the following sections, we briefly describe our approaches to answering *Rigid* and *Squishy* questions, and the results of our system using these approaches.

[*] now at: School of Computing Science, Simon Fraser University

[1] http://www.wjh.harvard.edu/~inquirer/

2 Answering Rigid Questions

Since *Rigid* questions can be seen as opinion-based *List* questions, we decided to keep the existing *List* question answering system as the foundation of our system. In this section, we will explain our approach to answering *List* questions, which is also used for *Rigid* questions. The description of this module can be found in more detail in [3] and [4].

Our *List* module is based on the observations that:

1. The instances of the answer to a *List* question have the same semantic entity class;
2. The instances of the answer tend to co-occur within the sentences of the documents related to the target and the question;
3. The sentences containing the answers share similar context.

We use the target and the question keywords as the representatives of context. In other words, we hypothesize that the instances of an answer to a *List* question tend to co-occur together and also tend to co-occur with the target and the question keywords within the sentences of the relevant documents. Co-occurrence can be an indicator of semantic similarity; generally, terms that co-occur more frequently tend to be related. The overall architecture of our *List* answerer subsystem is illustrated in Figure 1.

2.1 Candidate Answers Extraction

The first step to creating a list of candidate answers is answer type recognition. In addition to the nine classes defined for *List* questions: Person, Country, Organization, Job, Movie, Nationality, City, State, Other, two more classes are defined to deal with *Rigid* questions:

- Blog that refer to the name of the blog or its address to answer questions such as *What bloggers' sites contain negative opinions of Sean Hannity?* (sample question 9904.3)
- Blogger that refer to the author of the blog (appearing in patterns such as "posted by ") to answer questions such as *Who has expressed a negative opinion about the association of members of the FDA advisory committees?* (sample question 9906.2).

Each question is associated to one of the specified entity classes. This is done by using lexical and syntactic patterns. Once the type of answer is predicted, a number of documents are retrieved from Blog06 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 NE taggers (in case of Person, Organization and Job), using gazetteers (in case of Country, Nationality, State and partially City), by applying lexical patterns on the text of Blog06 documents (in case of Blog and Blogger), and finally by extracting all capitalized terms and terms in quotations and validating them using web frequency (in case of Movie and Other).

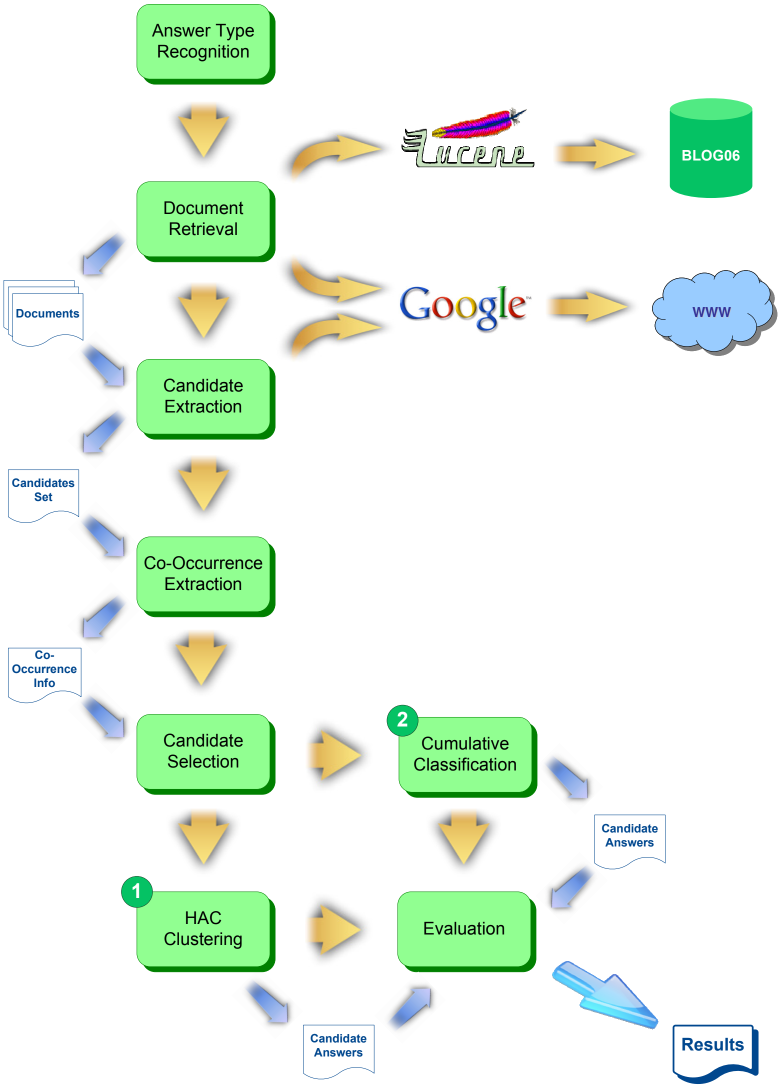

Figure 1: *List* Answering Subsystem Architecture as used for *Squishy* questions

2.2 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 BLOG06 and the web are retrieved. This constitutes the domain collection from which sentences will be extracted to compute co-occurrence information. For each pair of candidates, a 2×2 contingency table is created:

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.

The similarity between each pair of terms is computed using a modified version of weighted

	$\mathbf{term_i}$	$\neg$ $\mathbf{term_i}$	**Total**
$\mathbf{term_j}$	O_{11}	O_{21}	$O_{11} + O_{21}$
$\neg$ $\mathbf{term_j}$	O_{12}	O_{22}	$O_{12} + O_{22}$
[gray]0.7**Total**	$O_{11} + O_{12}$	$O_{21} + O_{22}$	N

Table 1: 2×2 Contingency Table Containing Co-Occurrence Information of Two Terms

Mutual Information. For more details about the modified weighted Mutual Information that we used, refer to [4].

$$\mathbf{WMI}\left(\mathbf{term_i},\, \mathbf{term_j}\right) = \begin{cases} -2 \times \dfrac{F_1 \times F_2}{N} \times \dfrac{\log_2 e}{e} & \text{if } O_{11} = 0 \\ -2 \times \dfrac{F_1 \times F_2}{N} \times \dfrac{\log_2 e}{e} + O_{11} \times \log_2 \dfrac{N \times O_{11}}{F_1 \times F_2} & \text{if } O_{11} < \dfrac{F_1 \times F_2}{N \times e} \\ O_{11} \times \log_2 \dfrac{N \times O_{11}}{F_1 \times F_2} & \text{if } O_{11} \geq \dfrac{F_1 \times F_2}{N \times e} \end{cases}$$

2.3 Candidate Answer Selection

In this section, we present and investigate two different approaches to returning a subset of candidate answers based on the co-occurrence information extracted in the previous stage. First, a clustering method is used to group closely related terms. Candidates in the most appropriate cluster are returned as the final candidates. Second, we use a simple classification method to classify the candidates into two classes: *Relevant* and *Irrelevant*. Then, the candidates in the class *Relevant* are returned.

2.3.1 Hierarchical Agglomerative Clustering

Since we do not have information regarding the coordinates of the candidate terms and the only information we have is the similarity between candidate terms, many clustering methods can not be applied. We use a Hierarchical Agglomerative clustering method which can be used in similar situations. The algorithm is as follows:

1. Remove terms that appear less than a certain threshold.

2. Sort the candidate answers based on their frequency in the domain collection.

3. Put each candidate term t_i in a separate cluster C_i.

4. Compute the similarity between each pair of clusters.
5. Merge two clusters that have the highest similarity between them.
6. Goto step 3 until there are only N clusters left or the highest similarity is less than a threshold.

After the clustering is finished, the main concern is how to select the right cluster. For this purpose, before clustering, we stem the target and question keywords and add them 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* (or *Rigid*) question tend to co-occur with one another and with the target and question keywords as well.

2.3.2 Cumulative Classification

We also experimented with a different method: *Cumulative Classification.* The classification is based on the same co-occurrence information used in clustering. The idea is that candidate terms that generally co-occur more often with other candidate answers, hence with a relatively higher similarity value, are more likely to be the answer. Therefore, we use the sum of the similarities of each term to other terms as an indicator of how often each term co-occurs with all other terms. Candidates with a cumulative similarity greater than a certain threshold are labelled as *Relevant* and the rest are *Irrelevant.* Table 2 illustrates how the cumulative similarity of each term to other terms is computed.

1. For each term in the candidate list, compute the sum of its similarities to all other candidate terms;
2. Sort the candidate terms based on their cumulative similarities;
3. Compute the threshold based on the above formula;
4. Retain only candidates whose cumulative scores are equal to or higher than the threshold;
5. Return the candidate terms as the final candidate terms.

2.4 Answer Projection

To support the final answers with documents from Blog06, a simple method is used. The corpus is searched using the query generated before and the final candidate term and simply the first document is returned as the supporting document. Several different approaches were examined that all caused a lower score than this method.

	$\mathbf{term_1}$	$\mathbf{term_2}$	$\cdots$	$\mathbf{term_n}$
$\mathbf{term_1}$	0	$Sim_{2,1}$	$\ldots$	$Sim_{n,1}$
$\mathbf{term_2}$	$Sim_{1,2}$	0	$\ldots$	$Sim_{n,2}$
$\vdots$	$\vdots$	$\vdots$	0	$\vdots$
$\mathbf{term_n}$	$Sim_{1,n}$	$Sim_{2,n}$	$\ldots$	0
Score	$\sum_{1\leq i\leq n} Sim_{1,i}$	$\sum_{1\leq i\leq n} Sim_{2,i}$	$\ldots$	$\sum_{1\leq i\leq n} Sim_{n,i}$

Table 2: Cumulative Similarity Table

3 Answering Squishy Questions

Since *Squishy* questions share similar characteristics with *Other* questions, we decided to exploit our existing *Other* question answering system as the underlying system for answering *Squishy* questions. However, the conventional *Other* questions ask for any other important and interesting piece of information about the target, while the new *Squishy* type of questions provides exactly what is considered interesting.

This section will present a brief summary on the approach we had used to answer *Other* questions, which we used with minor modifications to answer *Squishy* questions. For more details about how *Other* questions are answered, refer to [5] and [4].

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[2] online dictionary. The first stage to answering *Squishy* questions is to find the proper Wikipedia article.

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

We use the Google API to search in the Wikipedia domain using the target and the question 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 BLOG06 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.

4 Results

As mentionned in Section 1, due to technical problems and time constraints, we were unable to submit our main run in which we planned to use sentiment classification to answer opinion questions. Our results, therefore, only reflect the use of last year's system for *List* and *Other* questions used (almost) as is for the *Rigid* and *Squishy* questions. Our results turned out to be much lower than we had expected.

For the *Rigid* part, we used *Cumulative Classification* with modified weighted mutual information discussed in 2.3. Precision, recall and F-score of the answers are 0.104, 0.020 and 0.025 respectively. For the *Squishy* questions, our system got 0.028, 0.109 and 0.073 for precision, recall and F-score respectively.

5 Conclusion

In this paper, we described our approach to answering two new types of questions, *Rigid* and *Squishy*. Since we could not submit our main run which was supposed to reflect the modifications regarding these new question types, the results received are not representative of our approach. However, using the correct answers which will be published by TAC organizers, we can compare our sentiment approach to the approaches used by other participants. This needs to be done before any other future work.

References

[1] A. Fee M. Razmara and L. Kosseim. Concordia University at the TREC-2007 QA Track. In *Proceedings of the 16th Text Retrieval Conference (TREC-16)*, Gaithersburg, Maryland (USA), November 2007. TREC-2007.

[2] I. Ounis, M. de Rijke, C. Macdonald, G. A. Mishne, and I. Soboroff. Overview of the trec-2006 blog track. In *TREC 2006 Working Notes*, pages 15–27, November 2006.

[3] Majid Razmara and Leila Kosseim. Answering list questions using co-occurrence and clustering. In European Language Resources Association (ELRA), editor, *Proceedings of the Sixth International Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008.

[4] Majid Razmara. Answering *List* and *Other* questions. Master's thesis, Department of Computer Science and Software Engineering, Concordia University, Montreal, Canada, August 2008.

[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.