National University of Singapore at the TREC-13 Question Answering Main Task

Hang Cui Keya Li* Renxu Sun* Tat-Seng Chua Min-Yen Kan
Department of Computer Science
School of Computing
National University of Singapore
{cuihang, likeya, sunrenxu, chuats, kanmy}@comp.nus.edu.sg

1 Introduction

Our participation at TREC in the past two years (Yang *et al.,* 2002, 2003) has focused on incorporating external knowledge to boost document and passage retrieval performance in event-based open domain question answering (QA). Despite our previous successes, we have identified three weaknesses of our system with respect to this year's task guidelines. First, our system works at the surface level to extract answers, by picking the first occurrence of a string that matches the question target type from the highest ranked passage. As such, our answer extraction relies heavily on the results of passage retrieval and named entity tagging. However, a passage that contains the correct answer may contain other strings of the same target type (Light *et al.*, 2001), which means an incorrect string may be extracted. A technique to select the answer string that has the correct relationships with respect to the other words in the question is needed. Second, our definitional QA system utilizes manually constructed definition patterns. While these patterns are precise in selecting definition sentences, they are strict in matching (i.e., slot-by-slot matching using regular expressions), failing to match correct sentences with minor variations. Third, this year's guidelines state that factoid and list questions are not independent; instead, they are all related to given topics. Under such a contextual QA scenario, we need to revise our framework to exploit the existing topic-relevant knowledge in answering the questions.

Accordingly, we focus on the following three features in this year's TREC:

(1) To provide appropriate evidence for answer extraction, we use grammatical dependency relations among question terms to reinforce answer selection. In contrast to previous work in matching dependency relations, we propose measuring the similarity between relations to rank answer strings.

(2) To obtain higher recall in definition sentence retrieval, we adopt soft matching patterns (Cui *et al.*, 2004a). Unlike conventional lexico-syntactic patterns matched by regular expressions (*i.e.*, hard-matching patterns), soft patterns represent each slot as a vector of words and syntactic classes with their distributions, rather than generalizing specific instances. This allows us to probabilistically match test instances against the training data.

(3) To answer topically related factoid and list questions, we first combine sentences from our definition sentence retrieval module with downloaded definitions from external resources. This sentence base is used to answer factoid and list questions. Although using such a definition sentence base restricts recall in passage retrieval, it improves efficiency and effectiveness in answering common questions about people and organizations.

This paper is organized as follows: In the next section, we present the overall architecture of our system. In Sections 3, 4 and 5, we give the details of the above three features. In Section 6, we conclude the paper with future directions.

2 System Overview

In Figure 1, we illustrate the architecture of our QA system. We have leveraged our prior work in question analysis, document retrieval, query expansion and passage retrieval to build the system. In our comprehensive pre-processing step, we store a named entity profile and a full parse of each article in the TREC corpus. The offline processing greatly accelerates answer extraction.

Our framework functions as follows:

- **Target analysis and document retrieval:** First, the user submits a topic, *e.g.,* "Aaron Copland", to the system. Lucene[1] is used to index and retrieve the relevant documents. Topics are often coupled with qualifiers, for instance, "*skier* Alberto Tomba". We rely on the Web to separate the qualifiers from the main topic words, *e.g.,* "Alberto Tomba" in the above example.

---

* These two authors are listed in the alphabetical order of their last names.

[1] http://jakarta.apache.org/lucene/docs/index.html. Lucene performs Boolean search.

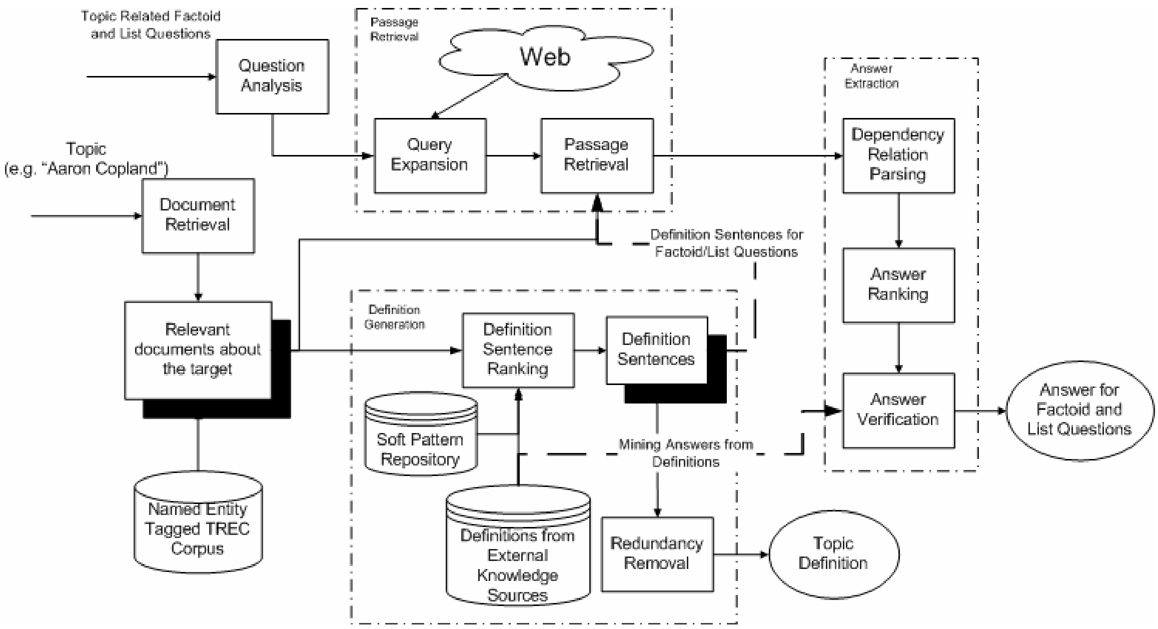

Figure 1. The illustration of the TREC QA system architecture

Specifically, we calculate the pointwise mutual information (PMI)[2] between each pair of topic terms based on the hits returned by Google when using the topic terms as query. Terms with PMI values beyond a pre-defined threshold are grouped together. To construct a suitable Lucene query, terms in the same group are first connected by "AND", and then different groups are connected by "OR". To handle errors or infrequent expressions in the given topics, we replace our original query by any query suggestion from Google[3]. For instance, our system automatically changes "Harlem Globe Trotters" to "Harlem GlobeTrotters" according to Google's result. From the document retrieval on the NE pre-tagged corpus, we get a set of NE tagged relevant documents related to the given topic.

- **Definition generation:** The relevant document set for the given topic is the basis for generating the definition for that topic. The definition generation module first extracts definition sentences from the document set. It identifies definition sentences using centroid-based weighting and definition pattern matching. It also leverages existing definitions from external resources. We discuss definition sentence extraction in Section 4. After redundancy removal, the module produces the definition for the topic.

- **Passage retrieval and query expansion for factoid and list questions:** To answer topically related factoid and list questions, we perform passage retrieval on two sources: the topic's relevant document set and the definition sentence set produced by the definition generation module. In our submissions to this year's TREC, the first and second runs for factoid questions used the whole relevant document set for passage retrieval; in the third run, we experimented with using only definition sentences to find answers to factoid questions.

We use a simple linear expansion strategy for query expansion. The method picks expansion terms from Google snippets according to the terms' co-occurrences with the question terms in the snippets. The passage retrieval module takes in expanded queries as input, and performs density-based lexical matching to rank passages, which consist of a window of three sentences. We list the detailed algorithm for passage retrieval in Appendix 1.

- **Answer extraction:** We perform rule-based question analysis to assign question target type to each question. Before question typing, we substitute the topics for all pronouns in the questions. For example, the question: "What is their gang color?", for the topic "Crips", is transformed into: "What is Crips' gang color?" This step facilitates dependency relation parsing in later steps. Highly ranked passages are fed into the answer extraction module. Both the question and candidate answer passages are parsed by MiniPar (Lin, 1998), a robust parser for grammatical dependency relations. The module ranks all possible strings of the appropriate type by how

---

[2] $PMI = \frac{P(X,Y)}{P(X)}$

[3] defined as when Google returns : "Did you mean: XXX"

closely they model relations to other question terms as encountered in training. We will discuss the ranking of answer strings using approximate dependency relation matching in the next section.

3 Approximate Dependency Relation Matching for Answer Extraction

By analyzing a subset of TREC-9 questions, Light *et al.* (2001) estimated an upper bound of 70% for the performance of a question answering system under the condition of perfect passage retrieval, named entity detection and question typing. Given the fact that there is always error in syntactic parsing and passage retrieval, the actual performance of answer extraction is worse. The ceiling on performance is created when many named entities of the same type appear close to each other, confusing answer selection. Without any knowledge of syntactic relations between the entities, a system might select the named entity nearest to the question terms. In addition, some questions, such as: "*What does AARP stand for?*" have no known named entity types to represent the question target. We believe the key to overcoming such linguistic ambiguity is to use deep syntactic analysis on both the question and answer text. To this end, we extract grammatical dependency relations between entities and use approximate matching of such relations in answer evaluation.

3.1 Extracting Dependency Relation Triples

Combining dependency relations in question answering is not a new idea. PIQASso (Attardi *et al.,* 2001) tested usage of syntactic relations generated by Minipar, a free robust dependency parser, in their QA system. However, their system produced low recall on the TREC data set, due to their use of keyword-based document retrieval (Katz and Lin, 2003). In contrast, Katz and Lin (2003) implemented a system to index and match all syntactic relations on the whole corpus. The weakness of existing systems that try to incorporate dependency parsing is that they use exact matching of relations to locate answers. Although such exact indexing and matching of relations result in high precision, they fare poorly in recall due to variations in both lexical tokens and syntactic trees.

Following the approaches taken by the existing work, we extract all relation path triples generated by the Minipar dependency parser from a given question and a candidate answer sentence. A *relation triple* is the smallest representation of a dependency path embedded in the parsing tree of a sentence. Each triple consists of two slots and one path of relations between them:

$$<Slot_1, Path, Slot_2>$$

where *slots* are either open-class words, like nouns and verbs, or named entities. A *path* represents the relation vector, consisting of a series of relations without taking their end nodes extracted from the parsing tree. For example, given the question: "What American revolutionary general turned over West Point to the British?" and answer sentence: "… Benedict Arnold's plot to surrender West Point to the British", we get the following triples[4]:

q1) General	*sub*	*obj*		West Point
q2) West Point	*mod*	*pcomp-n*		British
s1) Benedict Arnold	*poss*	*s*	*sobj*	West Point
s2) West Point	*mod*	*pcomp-n*		British

It is difficult to find identical relation structures between questions and answers. This is seen in the case above, where a correct answer is given but the relation structures differ. Although the triple (s2) matches the triple (q2) from the question, the string "Benedict Arnold" would not be selected as answer according to existing techniques because there is no match for the triple (q1). Approximate matching is needed to evaluate candidate answers. Clearly, we need a similarity measurement to represent how likely the two paths, namely "*sub obj*" and "*poss s sobj*", refer to the same relation chain.

3.2 Learning Relation Similarity

Common dependency relations are used interchangeably. Due to the variations existing in natural language text, the same relation may be phrased differently for questions and answer sentences. For instance, the appositive relation that appears frequently in news texts could correspond to other relations in a question. To obtain similarity measures among paths, we adopt a statistical method to learn the relatedness of relations from training data.

We accumulate around 1,000 factoid question-answer pairs from the past two years' TREC QA tasks to build our statistical model. We use Minipar to parse all the questions and their correspondent answer sentences. For each question-answer pair, relation paths from the question triples are aligned with those from the answer sentence if their slot fillers are the same after stemming. In order to get relations between answers and other question terms, we substitute a general tag for those question targets in questions

[4] We list only part of the extracted triples for the sake of space. A path exists between any pair of two open class words or named entities. We also restrict the length of the path to seven relations between the two slots.

and those answer strings in answer sentences. This results in 2,557 relation path pairs for model construction. The relatedness of two relations is measured by their co-occurrences in both question relation paths and answer relation paths. We employ a variation of mutual information to represent relation co-occurrences. Unlike normal mutual information, we account for path length in our calculation. Specifically, we discount the co-occurrence of two relations in long paths. The mutual information is presented as:

$$MI(\mathrm{Re}l_0, \mathrm{Re}l_1) = \log \frac{\sum \alpha \times \delta(\mathrm{Re}l_0, \mathrm{Re}l_1)}{f_Q(\mathrm{Re}l_0) \times f_A(\mathrm{Re}l_1)} \quad (1)$$

where Rel_0 and Rel_1 are two relations extracted from question paths and answer paths respectively. $f_Q(Rel)$ and $f_A(Rel)$ represent the number of occurrences of Rel in question paths and answer paths. $\delta(\mathrm{Re}l_0, \mathrm{Re}l_1)$ is 1 when the relations Rel_0 and Rel_1 occur in a question path and its corresponding answer path respectively, and 0 otherwise. α is the inverse proportion of the lengths of the question path and the answer path.

We calculate pairwise similarity for all dependency relations based on this equation. These relation similarities form the basis for calculating relation path similarity in the evaluation of answer strings. Figure 2 shows an excerpt of the similarity measures between different relations.

Relation-1	Relation-2	Similarity
whn	*pcomp-n*	0.43
whn	*i*	0.42
i	*pcomp-n*	0.39
i	*s*	0.37
pred	*mod*	0.37
appo	*vrel*	0.35
whn	*nn*	0.34
s	*num*	0.33

Figure 2. Excerpt of similarity measures between relations

3.3 Evaluating Answer Strings

To ensure high recall, we feed the top 50 sentences from the passage retrieval module into the answer evaluation module. We consider two issues in selecting the correct answer: the correct named entity type as determined by question typing, and the similarity of paths between candidate answers and question terms in the question and the candidate answer sentence. For questions with an unknown target type, we examine all noun and/or verb phrases in the given sentences. We first align the relation paths anchored by matched question terms from the question and the answer sentence. We then combine the similarities of all relation paths. We rank the candidate answers by:

$$Weight(Ans) = \sum_{path} Sim(P^Q_{(Ans,*)}, P^A_{(Ans,*)}) \quad (2)$$

Here, P refers to all paths in the question or a candidate sentence with one slot being the question target or a candidate answer.

Recall that a relation path consists of several relations along the path in the parsing tree. To measure the similarity of two relation paths, we combine the similarities between their relations. In our submissions, we experimented with two different methods in aligning relations when calculating path similarities.

First, we treat relations along a path as a sequence of tokens and consider all possible alignments of relations between two paths without actually aligning any relations. We term this *total path matching,* which is similar to IBM's Model 1 statistical translation model (Brown *et al.* 1993). In our case, we use simple mutual information to represent their "translation probability". The path similarity is calculated by:

$$Sim(P_Q, P_A) = \frac{\varepsilon}{(1 + len(P_Q))^{len(P_A)}} \prod_j \sum_i MI(\mathrm{Re}l_i^Q, \mathrm{Re}l_j^A) \quad (3)$$

In the second configuration, which is called *relation triple matching*, we count only the similarities of individual relations that have the same slot fillers. In other words, only the relations between adjacent nodes that contain the question terms in the parsing tree are considered in the path similarity calculation. In this case, the alignments of relations are judged by their two end slot fillers. We combine all similarities of matched triples to rank candidate answers:

$$Sim(P_Q, P_A) = \frac{\varepsilon}{(1 + len(P_Q))^{N(M)}} \prod_{j \in M} MI(\mathrm{Re}l_i^Q, \mathrm{Re}l_j^A) \quad (4)$$

where M represents the set which contains all matched triples.

After ranking candidate answers by Equation 2, we select the highest ranked answer string, which has the appropriate target type and also falls into the verification list, as the final answer.

3.4 Evaluation Results and Discussions

We submitted three runs for factoid questions, all employing approximate dependency relation matching in answer extraction. The highest average accuracy of 0.625 was achieved by the configuration that used total path matching. The performance obtained by relation triple matching (average accuracy of 0.600) was close to it. We note that using triple matching to align

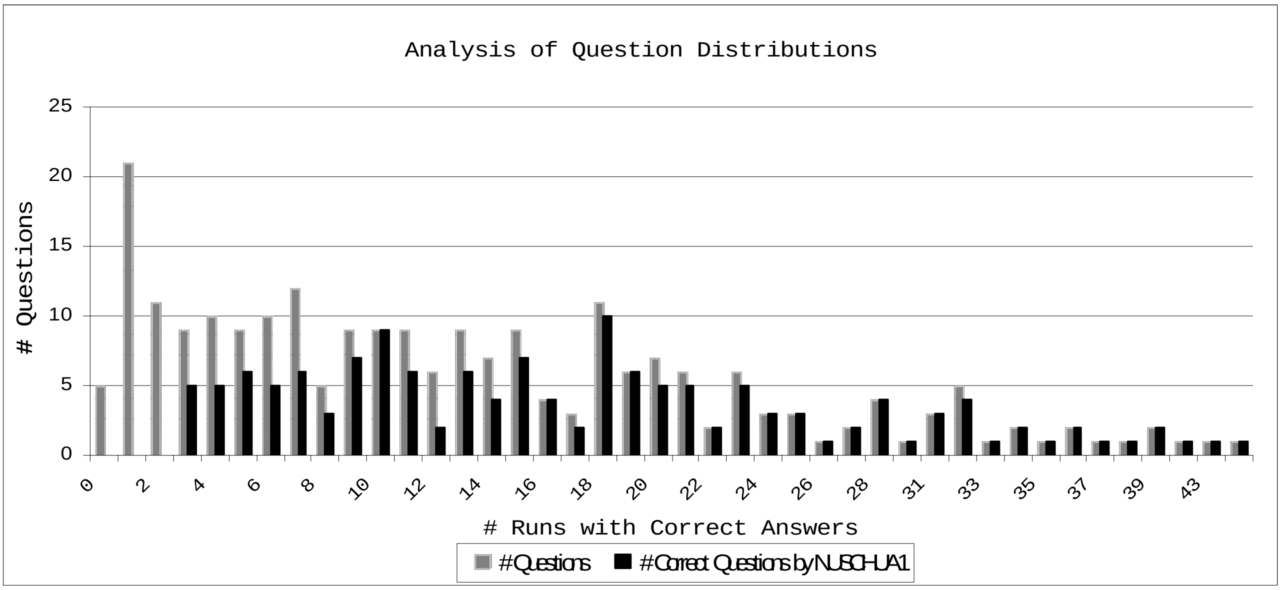

Figure 3. Illustration of distribution of questions

relations may mean deliberately ignoring the long dependency relationship between entities. However, we conjecture that this does not significantly degrade performance because dependency parsers may not resolve long distance dependency relations well.

Further examination reveals that our new method of measuring dependency relation path similarity in answer extraction outperforms our previous system, in which the first occurrence of a named entity with the correct type is returned for questions with a known answer type.

For all non-NE questions (in which the system is unsure of the question target type), the module picks the most probable noun phrase which is nearest to all question terms in the top ranked passage. These non-NE questions account for 69 out of the total 230 factoid questions according to our question typing module. We used our previous system as the baseline and compared it with the new answer extraction module in the first two runs we submitted. We list the results in Table 1. The table shows that leveraging more syntactic relations boosts the performance of answer string selection, especially where non-NE answers are involved.

We also analyzed the distribution of this year's factoid questions. We illustrate the distribution of questions according to the number of runs that gave the correct answers in Figure 3. We include the number of questions that were answered correctly by NUSCHUA1 in the figure as well. The X axis in Figure 3 stands for the number of runs with correct answers for the corresponding number of questions (as showed by axis Y) in all submitted runs to TREC. The left end of the X axis represents that no runs gave correct answers to the questions. As illustrated in Figure 3, our system does not perform well in answering difficult questions. As illustrated in the figure, it misses all questions that are correctly answered by one or two runs. This shows that although we have improved on our previous system by incorporating more complicated relation matching techniques, the system still has much room for improvement. One serious problem is the lexical gap, *i.e.,* the difference in vocabulary used to express the questions and those used in the passages. Our relation matching is conducted only when some question words are matched in the candidate passages. In future work, we may incorporate approximate matching of question terms in relation matching.

Table 1. Performance comparison of two submitted runs

	Baseline	NUSCHUA1	NUSCHUA2
Overall average accuracy	0.51	0.62	0.60
For questions with NE typed targets	0.68	0.78	0.75
For questions without NE typed targets	0.29	0.42	0.41

4 Definition Generation for Topics

To facilitate the answering of topic-related factoid and list questions as well as to provide sentences for answering Other questions, we deem it important to identify precise and complete definition sentences for the given topics. In last year's TREC definitional QA task, top ranked groups utilized a relatively uniform architecture for extracting definition sentences: (1) finding additional information for the topics from external web sites or thesauri; and (2) employing manually constructed definition patterns to identify sentences. Enlightened by our previous experimental results (Cui *et al.,* 2004b), we try to improve our previous system by using: (1) existing definitions from specific web sites, rather than generic web search; and (2) machine learned soft matching definition patterns, instead of manually constructed hard matching patterns represented in regular expressions. We combine the use of these two techniques to identify precise definition sentences.

4.1 Statistical Ranking of Definition Sentences with External Knowledge

To ensure recall, for each topic, we construct two data sets as the basis for selecting definition sentences: one based on the TREC corpus and the other from external knowledge. The TREC set is constructed by relevant documents determined by the document retrieval module using the topic as the query. We retrieve up to 800 documents for each topic. These documents are split into sentences. To construct the external knowledge set, we accumulate existing definitions for the topics from six specific web sites and glossaries. The external resources and their coverage of topics are listed in Table 2. The definitions are downloaded through pre-written wrappers for these sources. As Biography.com and S9 are dedicated biographical web sites, we do not search for definitions of organizations and other objects at these two sites.

Table 2. List of external resources for definitions and their coverage of topics.

External Resource Names	Coverage of Topics (out of 65 topics)
Biography.com (http://www.biography.com/)	19
S9 (http://s9.com/biography/index.html)	15
Wikipedia (http://en.wikipedia.org/wiki/Main_Page)	63
Bartleby.com (http://www.bartleby.com/)	37
Google Glossary (search by "define: <term>" in Google)	25
WordNet Glossary	13

We first perform statistical weighting of sentences on both of the data sets to find the sentences relevant to the given topics. When ranking sentences with corpus word statistics, we employ the centroid-based ranking method, which has been used in other definitional QA systems (*e.g.,* Xu *et al.,* 2003). We select a set of *centroid words* (excluding stop words) which co-occur frequently with the search target in the input sentences. To select centroid words, we use mutual information to measure the centroid weight of a word *w* as follows:

$$Weight_{centroid}(w) = \frac{\log(Co(w, sch\_term)+1)}{\log(sf(w)+1)+\log(sf(sch\_term)+1)} \times idf(w) \quad (5)$$

where *Co(w, sch_term)* denotes the number of sentences where *w* co-occurs with the search term *sch_term*, and *sf(w)* gives the number of sentences containing the word *w*. We also use the inverse document frequency of *w*, *idf(w)* [5], as a measurement of the global importance of the word. Words whose centroid weights exceed the average plus a standard deviation are selected as centroid words.

The weighting of centroid words can be improved by using external knowledge. We augment the weight of the centroid words which also appear in the definitions from the external knowledge data set. We form centroid words into a centroid vector, which is then used to rank input sentences by their cosine similarity with the vector.

4.2 Soft Matching Definition Patterns

By doing statistical ranking, we obtain a list of highly ranked sentences that are potential definition sentences. These sentences are closely relevant to the given topic but may not be necessarily definition sentences. Definition sentences, such as "Gunter Blobel, a molecular biologist …" are often written in a certain style or pattern.

Definition patterns in most TREC systems are manually constructed, which is labor intensive. These patterns are usually represented and matched using regular expressions. We consider these techniques *hard matching* because they require definition sentences to match exactly. The use of hard pattern rules fails to capture the variations in vocabulary and syntax that are often exhibited in definitions sentences; the method also cannot recognize definition patterns which are not explicitly found in training. To overcome this problem, we have proposed a probabilistic soft matching technique which computes the degree of match between test sentences and training instances (Cui *et al.,* 2004a). Given a set of training instances, a virtual vector representing the soft definition pattern *Pa* is generated by aligning

[5] We use the statistics from the Web Term Document Frequency and Rank site (http://elib.cs.berkeley.edu/docfreq/) to approximate words' IDF.

the training instances according to the positions of <SCH_TERM>:

<$Slot_{-w}$, ... $Slot_{-2}$, $Slot_{-1}$, SCH_TERM , $Slot_1$, $Slot_2$, ... $Slot_w$: Pa>

where $Slot_i$ contains a vector of tokens with their probabilities of occurrence derived from the training instances.

The test sentences are first preprocessed in a manner similar to the preprocessing of labeled definition sentences. Using the same window size *w*, the token fragment *S* surrounding the <SCH_TERM> is retrieved:

<$token_{-w}$, ..., $token_{-2}$, $token_{-1}$, SCH_TERM, $token_1$, $token_2$, ... $token_w$: S>

The degree of match between the test sentence and the generalized definition patterns is measured by the similarity between the vector *S* and the virtual soft pattern vector *Pa*, which accounts for similarity of individual slots as well as the sequence of slots. Our soft matching technique is described in detail in Cui *et al.* (2004a).

4.3 Manually Constructed Patterns

In addition to centroid-based weighting and soft pattern matching, we also use a set of manually constructed definition patterns, which is a subset of patterns we used for last year's TREC definitional QA task. These patterns, mainly consisting of appositives and copulas, are high-precision patterns represented in regular expressions, for instance "<SEARCH_TERM> is DT$ NNP". The purpose of using such hard matching patterns in addition to soft matching patterns is to capture those well-formed definition sentences that are missed due to the imposed cut-off of ranking scores by soft pattern matching and centroid-based weighting.

Therefore, the system works in stages: it ranks all sentences using centroid-based ranking and soft pattern matching, and takes the top ranked sentences as candidate definition sentences. It then examines those lower ranked sentences which are not included in the candidate definition sentences and adds in those sentences matched by any of the manually constructed patterns. In this way, we boost the recall of definition sentences identified by the sentence extraction module.

4.4 Redundancy Removal and Answer String Extraction

As the TREC QA guideline suggests, to answer Other questions, the nuggets that have been covered by those topic-related factoid/list questions are to be removed. Our system performs a two-stage redundancy check when selecting definition sentences into the final answer. Suppose we are to select *N* sentences for the final answer, the selection process works as follows:

a) Add the first sentence into the answer.
b) Examine the next sentence *next_stc*
   if $\max_i(sim(next\_stc, answer\_stc_i)) \geq 0.7$ continue;
   if $\max_j(sim(next\_stc, factoid\_stc_j)) \geq 0.85$ continue;
   else add *next_stc* into the answer;
c) Go to step (b) until *N* sentences have been selected.

Here, *answer_stc* refers to those sentences that have been previously selected as part of the answer for Other questions. *Factoid_stc* refers to those sentences that produce the answers for those factoid or list questions. We measure the similarity between two sentences using the simple cosine similarity which weights unigrams by their inverse document frequency (IDF). We apply a stricter similarity threshold for sentences used to answer factoid/list questions as the answers to such questions tend to amount for very small portion of the sentences.

In addition to full definition sentences, we also develop a set of heuristic rules to extract fragments from sentences in order to shorten the final answers. These heuristic rules are adopted from the system we developed last year. For instance, for a definition sentence that contains the appositive of the topic, only the appositive part is extracted. To avoid introducing confusion, all starting topic words of each sentence are also removed. For example, "TB, also known as tuberculosis …" is transformed into "also known as tuberculosis …"

4.5 Evaluation Results

We submitted three runs for Other questions. The runs differed only in the length of the cut-off criterion applied. The summary of the three runs is listed in Table 3.

Our second run achieved the highest score in the Other task. Due to the change of the F_5 measure to the F_3 measure, the length of the answers plays a more important role in the evaluations. It is crucial for us to develop a more systematic method for selecting definition answers than the current heuristics that we employ.

This year's Other task cannot be considered identical to last year's definitional QA task because it requires us to exclude all nuggets that have been covered by the topic-related factoid/list questions. This makes the evaluation of the Other task more difficult. Based on our observation, the essential aspects about a topic have been covered by the factoid/list questions. For instance, given a query on a singer, questions about standard topics of interest (such as his/her birthday, songs and band) are already posed through specific factoid or list questions. Thus, it is very difficult for a system to determine what "other" information is most important about the topic. We believe that this is the main reason that causes the overall scores of this task to decline.

Table 3. Summary of submitted runs for Other questions.

Runs	Answer string extraction applied	Average length (in bytes)	Final F_3 score
NUSCHUA1	No	2079	0.448
NUSCHUA2	Yes	1973	0.460
NUSCHUA3	Yes	2505	0.379

5 Exploiting Definitions to Answer Factoid/List Questions

We also experimented with using definition sentences to answer topic-related factoid/list questions. Our third run for factoid question answering illustrates such an idea. In this run, the definition sentence extraction module sent its top-ranked sentences to the passage retrieval module. The passage retrieval module ranked these definition sentences according to specific factoid/list questions. This approach, while efficient and effective in extracting answers to common questions about persons and organizations, tended to miss peripherally relevant passages. This run achieved an average precision of 0.50, which was lower than the runs that used the whole relevant document for passage retrieval. We conjecture that the cut-off threshold of selecting definition sentences leads to lower recall that affects passage retrieval.

We also incorporate existing definitions from external web sites and thesauri in answering certain types of list questions. Specifically, we utilize a set of manually constructed wrappers to acquire certain aspects of a person or a corporation. One of the wrappers is for extracting the names of a person's works, including his/her songs, movies, books and plays, which are often listed in a specific format in web sites. In this way, we can obtain a list of names or works directly from these sites. In addition, such lists of works are often presented in a uniform manner: they are often enclosed by quotation marks and consist of several capitalized words. Although these extracted lists may contain noise, false matches can be discarded by validating the list against existing definitions. As such, we have achieved high precision and recall for the eight list questions on people's songs, albums and books, with an average F measure of 0.81 and 0.73 respectively for the two runs. In addition to works, we have also pre-compiled a list of structured patterns for extracting product names of a company and working positions for a person. In future work, we plan to extend our soft matching patterns to accomplish this task to handle variations in news articles.

In addition, we have found that many fields of simple facts about a person can be extracted directly from existing definitions, such as birth/death date, birthplace and career. We believe that developing such a set of wrappers to mine simple facts would improve both the effectiveness and efficiency of the QA system.

6 Conclusion

We have reviewed the newly-adopted techniques in our QA system. They include measuring relation path similarity in answer extraction, soft matching patterns for identifying definition sentences, and using definitions about topics to answer topic-related factoid/list questions. While these techniques have improved our previous QA system, we note that more improvements may be pursued in future work. First, the mismatch of question terms is still a serious problem. It is crucial to devise a framework that can align semantically related words and calculate relation path similarities. Second, a generic method for selecting appropriate text fragments from definition sentences is necessary. The main challenge here is to identify relevant parts of the definition sentence when the match is only partial. Third, the performance gain obtained using definitions to answer common questions about a person or an organization still remains to be explored. More experiments should be conducted to figure out what kind of specific questions can be correctly answered by automatically generated and manually constructed definitions.

7 Acknowledgement

The authors are grateful to Shi-Yong Neo, Victor Goh and Yee-Fan Tan for their help with migrating the previous year's subsystems. We also thank Hui Yang for sharing her experience in participating in TREC QA. Thanks also go to Alexia Leong for proofreading this paper. The first author is supported by the Singapore Millennium Foundation Scholarship (Ref No: 2003-SMS-0230).

References

[Attardi *et al.,* 2001] G. Attardi, A. Cisternino, F. Formica, M. Simi and A. Tommasi, *PiQASso: Pisa Question Answering System*, Proceedings of text Retrieval Conference (Trec-10), 599-607, NIST, Gaithersburg(MD), November 13-16, 2001.

[Brown *et al.*, 1993] P. Brown, S. Della, V. Della Pietra and R. Mercer, *The mathematics of statistical machine translation: Parameter estimation*, Computational Linguistics, 19(2), pp. 263-311, 1993.

[Cui *et.al.,* 2004a] H. Cui, M.-Y. Kan and T.-S. Chua, *Unsupervised learning of soft patterns for definitional question answering,* Proceedings of the Thirteenth World Wide Web Conference (WWW 2004), New York, May 17-22, 2004, pp. 90-99.

[Cui *et.al.,* 2004b] H. Cui, M.-Y. Kan, T.-S. Chua and J. Xiao, *A Comparative Study on Sentence Retrieval for Definitional Question Answering,* Proceedings of SIGIR Workshop on Information Retrieval for Question Answering, Sheffield, U.K., 2004.

[Katz and Lin, 2003] B. Katz and J. Lin, *Selectively Using Relations to Improve Precision in Question Answering*, Proceedings of the EACL-2003 Workshop on Natural Language Processing for Question Answering, April 2003.

[Lee *et al.*, 2001] G. G. Lee, J. Seo, S. Lee, H. Jung, B.-H. Cho, C. Lee, B.-K. Kwak, J. Cha, D. Kim, J. An, H. Kim, and K. Kim, *SiteQ: Engineering high performance QA system using lexico-semantic pattern matching and shallow NLP*, In Proceedings of the Tenth Text REtrieval Conference (TREC-10), 2001, pp. 442-451.

[Light *et al.,* 2001] M. Light, G. Mann, E. Riloff and E. Breck, *Analysis for elucidating current question answering technology*, Journal of Natural Language Engineering, Fall-Winter, 2001.

[Lin, 1998] D. Lin, *Dependency-based Evaluation of MINIPAR,* In Workshop on the Evaluation of Parsing Systems, Granada, Spain, May, 1998.

[Xu *et al.,* 2003] J. Xu, A. Licuanan, R. Weischedel, *TREC 2003 QA at BBN: Answering Definitional Questions*, The Twelfth Text REtrieval Conference (TREC 2003) Notebook, pp. 28-35, 2003.

[Yang *et al.,* 2003] H. Yang, H. Cui, M.-Y. Kan, M. Maslennikov, L. Qiu and T.-S. Chua, *QUALIFIER in TREC-12 QA Main Task*, In the notebook of the 12th Text REtrieval Conference (TREC'2003), Maryland, USA.

[Yang *et al.,* 2002] H. Yang and T.-S. Chua, *The Integration of Lexical Knowledge and External Resources for Question Answering*, In the Proceedings of the Eleventh Text REtrieval Conference (TREC'2002), Maryland, USA, 19-22 Nov 2002, pp. 155-161.

8 Appendix

Appendix 1. Algorithms for Passage Retrieval

We employ a density-based passage retrieval algorithm that is reinforced by query expansion. To select expansion terms for a factoid or list question, we submit the question to Google and select the top 10 weighted terms from returned snippets as expansion terms based on the following weighting scheme:

$$Weight\_Exp(t) = IDF(t) \times \log_{10}(|\,snippets(t)\,|) \qquad \text{(A1)}$$

where *IDF(t)* denotes the inverse document frequency of term *t* and |*snippets(t)*| gives the number of snippets that contain *t.* As we filter out those snippets that do not contain any target term, the counting of snippets is equivalent to the co-occurrence of term *t* with the target.

We take sentences as passages. The reason is that we employ Minipar in final answer extraction and Minipar can only resolve dependency relations within a sentence. We weight a sentence as the combination of three partial scores: word overlap, word density and weights of its adjacent sentences. In other words, a sentence's weight could be augmented by adjacent sentences that have high weight. It is similar to the idea used in SiteQ (Lee *et al.,* 2001) that rank *n*-sentence passages. In particular, we weight sentence S_i as follows:

$$Weight(S_i) = (W_{overlap} + W_{Density}) \times \eta + \Delta Weight(S_i) \qquad \text{(A2)}$$

where

$$W_{overlap} = \sum_{i \in QT} IDF(t_i) + \sum_{k \in EXP} IDF(t_k) \qquad \text{(A2.1)}$$

$$W_{Density} = W_{Density}^{(QT)} + W_{Density}^{(EXP)} \qquad \text{(A2.2)}$$

$$W_{Density}^{(QT)} = \frac{\sum_{j=1}^{|QT|-1} \frac{IDF(t_j) + IDF(t_{j+1})}{\alpha \times dist(j, j+1)^2}}{|\,QT\,| - 1} \times |\,matched\_qt\,| \qquad \text{(A2.3)}$$

$$W_{Density}^{(EXP)} = \frac{\sum_{t=1}^{|EXP|-1} IDF(t_t) + IDF(t_{t+1})}{\beta \times dist(t, t+1)^2} \times |\,matched\_\exp t\,| \qquad \text{(A2.4)}$$

$$\Delta Weight(S_i) = \begin{cases} 2 \times \delta \times S_{i+1} & first\_sentence(i=1) \\ 2 \times \delta \times S_{i-1} & last\_sentence(i=N) \\ \delta \times (S_{i+1} + S_{i-1}) & otherwise \end{cases} \qquad \text{(A2.5)}$$

$$\eta = \begin{cases} 0.2 & \text{if } length(S_i) > 500 \text{ bytes} \quad or \quad length(S_i) < 30 \text{ bytes} \\ 1 & otherwise \end{cases}$$

(A2.6)

QT – non-trivial question terms
EXP – expansion terms
matched_qt – matched question terms in the passage
matched_expt – matched expansion terms in the passage
α, β, δ – constants