Using Syntactic and Semantic Relation Analysis in Question Answering

Renxu Sun Jing Jiang Yee Fan Tan Hang Cui Tat-Seng Chua Min-Yen Kan

Department of Computer Science
School of Computing
National University of Singapore
{sunrenxu,jingjian,tanyeefa,cuihang,chuats,kanmy}@comp.nus.edu.sg

1 Introduction

Our participation at TREC this year focuses on integrating dependency and semantic relation analysis of external resources into our existing QA system. In TREC-13, we have proposed the use of dependency relation matching to perform answer extraction for factoid and list questions. The results showed that the technique is effective in answer extraction within the corpus. However, we have also identified some problems and limitations with this technique. First, dependency relation matching does not perform well on short questions, which have only very few key terms. Therefore we need to integrate query expansion and make use of external resources to provide additional contextual information to these short questions. Second, the technique cannot be directly applied to extract answer nuggets from external web pages. As web pages contain much more noise as compared to the corpus, statistical based dependency relation matching tends to make a lot of errors based on our previously trained model on the corpus. Moreover, we do not have sufficient training data to retrain a model on the web. Therefore we propose to use semantic relation analysis to supplement dependency relation analysis to extract answer nuggets for factoid and list questions on the web. Finally, we adopt a soft pattern matching model (Cui *et al.,* 2005) for definition sentence retrieval in the definitional QA task.

We focus on the following three features in this year's TREC:

(1) We incorporate dependency relation analysis into query expansion based on external web resources, and we perform query expansion for both document and passage retrieval to provide more contextual information for short questions.
(2) We propose the use of semantic relation analysis to extract answer nuggets for factoid and list questions from external web resources. These external answer nuggets can either be directly projected back to the corpus (for list questions) or be used as additional supporting evidence for corpus-based answer extraction (for factoid questions).
(3) We train a bigram soft pattern statistical model to capture the patterns inherent in a set of training examples. We employ the bigram model to identify good definition sentence candidates.

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 respectively give the details of the above three features. In Section 6, we conclude the paper with discussion of 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, passage retrieval to build the system. Our major modification lies in query expansion, semantic answer nugget extraction from web resources and the bigram model for definition sentence selection. In our comprehensive pre-processing step, we store a named entity profile and the full parsing 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 the documents. In handling topics 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. 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, we first use logical "AND" to connect terms in the same group, and employ logical "OR" to connect different groups. To handle errors or infrequent expressions

[1] http://jakarta.apache.org/lucene/docs/index.html

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

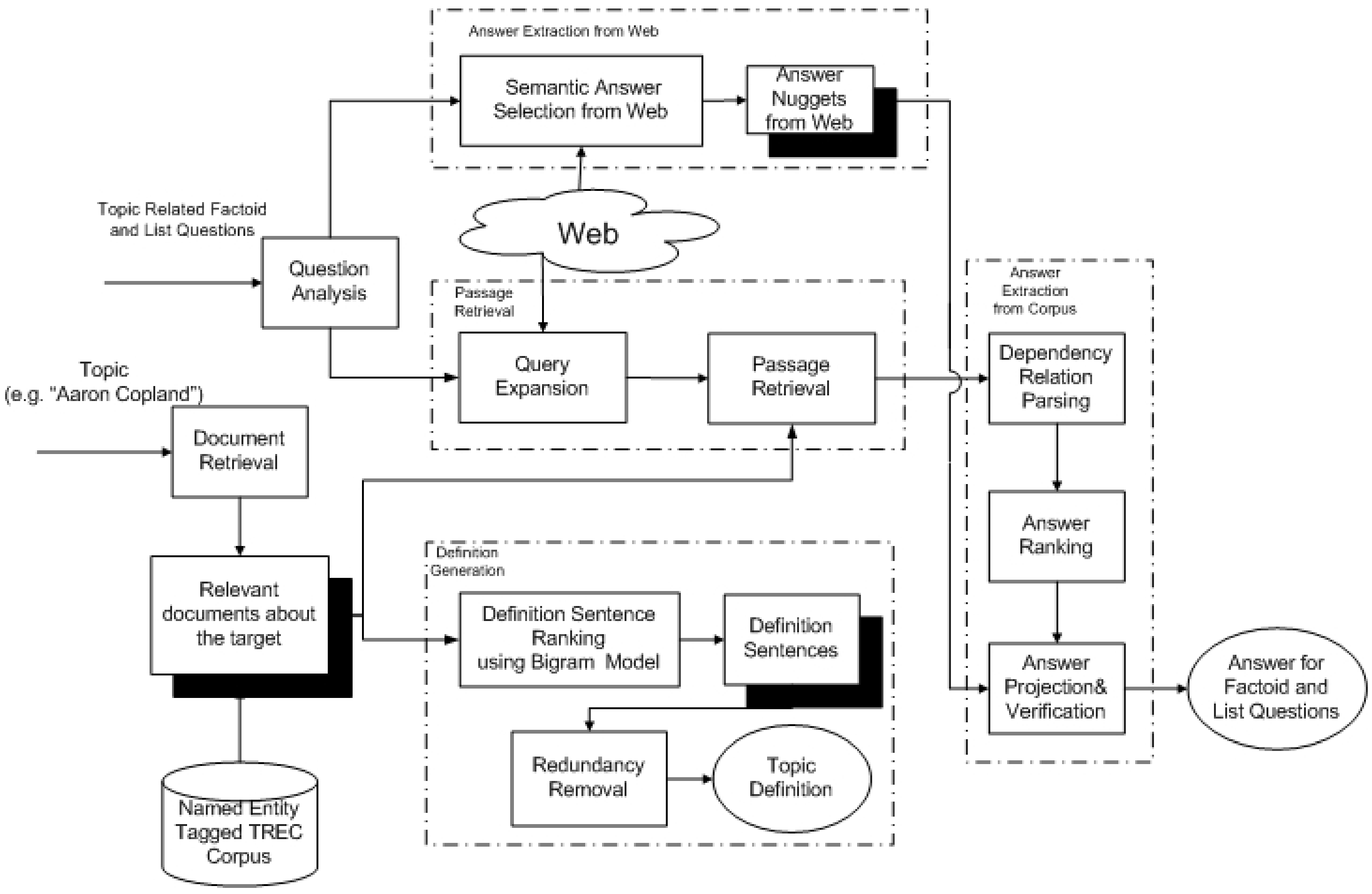
Answer Extraction from Web
Semantic Answer Selection from Web
Answer Nuggets from Web
Web
Topic Related Factoid and List Questions
Question Analysis
Passage Retrieval
Query Expansion
Passage Retrieval
Answer Extraction from Corpus
Dependency Relation Parsing
Answer Ranking
Answer Projection& Verification
Answer for Factoid and List Questions
Topic (e.g. "Aaron Copland")
Document Retrieval
Relevant documents about the target
Definition Generation
Definition Sentence Ranking using Bigram Model
Definition Sentences
Redundancy Removal
Topic Definition
Named Entity Tagged TREC Corpus

Figure 1. The illustration of the TREC QA system architecture

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.

- **Factoid/List Question Analysis:** We first extract the expected answer NE type for each question. We then parse each question using Minipar (Lin, 1998) and store the dependency parse tree, which will be used for dependency based answer extraction. Finally, we parse the question using shallow semantic parser ASSERT (Pradhan *et al.,* 2004) and also store the parse tree, which will be used for semantic answer nugget extraction from web resources. As some of the questions cannot be parsed by ASSERT (i.e empty output), we only perform semantic parsing on the subset of questions which have non-empty output. For the rest of the questions, we only perform dependency based answer extraction.

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

- **Query expansion and passage retrieval for factoid and list questions:** We incorporate dependency relation analysis into query expansion, which will be introduced in Section 3. The method picks expansion terms from Google snippets according to the terms' relation with the question terms in the snippets. For document ranking task this year, we select the top k expanded terms together with the non-trivial question terms to form the query. Our passage retrieval module also takes in expanded queries as input, and performs density-based lexical matching to rank passages, which consist of a window of three sentences.

- **Answer extraction:** We perform answer extraction on corpus documents as well as answer nuggets extraction on external web documents and select the final answer by answer projection (for list questions) and verification (for factoid questions). We use dependency based answer extraction to extract the answer string from the corpus. However, our preliminary experiment shows that this approach does not work well on web pages as they contain much more noisy data as compared to corpus. Therefore we propose the use of semantic based answer nugget extraction which is less sensitive to noise and we give the technical details in Section 4.

- **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 then applies the soft-pattern model for matching these definition sentences. It also leverages existing definitions from external resources. We will discuss definition sentence extraction in Section 5. After redundancy removal, the module produces the definition for the topic.

3 Query Expansion using Dependency Relations

In TREC-13, we have proposed to use dependency relation matching to perform answer selection. However, our experiments showed that dependency relation matching does not perform well on short questions with very few (less than four) key words. Therefore we need to introduce additional contextual information for these short questions through query expansion. However, most query expansion methods only introduce new terms and cannot be directly applied to relation matching. Thus we propose a query expansion algorithm, which can expand new terms as well as relation paths based dependency relation analysis.

To perform query expansion, we first send the queries to Google and use the top 50 returned snippets as a basis for query expansion. We parse the snippets using MiniPar and rank each non-stop token in the parsing tree of the snippet by its relation path to the tokens in the parsing tree of the question using the trained scores of individual relation. Finally, we select the top k relation paths in the snippets to combine with relation paths derived from the original questions to perform answer selection.

We will next introduce our query expansion algorithm follow by details on how we train individual relation scores.

3.1 Query Expansion

Most query expansion techniques rank expansion terms using their co-occurrence with the query term by performing local context analysis (Xu *et al.*,1996). However, we observed that due to the noise on the web, the same technique cannot be applied to the Web. This is because some irrelevant terms, such as commercial related terms co-occur very frequently with the query terms in some snippets. This will mislead the query expansion algorithm to select them as relevant terms. Therefore we propose to use dependency relation between query terms and expanded terms as additional evidence to infer the relevance of the expanded term. Our general framework is similar to the local context analysis method, but with two major differences: (1) We perform query expansion using web resources rather than the top N passages retrieved within the corpus. (2) We score the expanded terms using their relation paths to query terms rather than statistical co-occurrence.

Below are the steps we use to incorporate dependency relation analysis to expand a query Q based on web resource D_w.

1) We input each question as a query to Google and collect the top 50 returned snippets as a basis for query expansion. We combine the 50 snippets as a whole document denoted as D_w and perform sentence splitting and dependency parsing using Minipar. Each sentence S_i in D_w becomes a dependency tree T_i after parsing. A dependency tree depicts the dependency relations between tokens of a sentence. For any two tokens (a token may either be a single word, a noun phrase or a verb phrase) in a sentence, there exists a path between them. The path consists of a series of intermediate nodes linked by labeled edges called relations. So we can define relation path in the form of (Start_Token, Rel_1… Rel_k…Rel_m, End_Token).
2) After step 1, we have *N* dependency parsing trees that corresponds to *N* sentences in D_w. The non-stop word tokens denoted as Tk in D_w are ranked according to the formula:

$$Score(Tk,Q)=\prod_{t_i\in Q}\left(\frac{\delta+\log_{10}(\sum_{s=1}^{n} path\_score(Tk,t_i,s))\times idf_{Tk}}{\log_{10} N}\right)^{idf_{t_i}} \quad (3.1.1)$$

Where

$$path\_score(Tk,t,s)=\prod_{\substack{Tk\in s\wedge t\in s\wedge\\ \mathrm{Re}\,l_i\in path(Tk,t)}} score(\mathrm{Re}\,l_i) \quad (3.1.2)$$

$$idf_{Tk}=\log_{10}(N/N_{Tk})/\log_{10} N \quad (3.1.3)$$

$$idf_{t_i}=\log_{10}(N/N_{t_i})/\log_{10} N \quad (3.1.4)$$

Tk is the token to be ranked,

$path(Tk,t,s)$ is all the relation paths in the dependency parsing tree of sentence s with start token Tk and end token t,

$path\_score(Tk,t,s)$ is the score of $path(Tk,t,s)$

N is the number of sentences in D_w,

N_{Tk} is the number of sentences in D_w contains Tk,

N_{t_i} is the number of sentences in D_w contains t_i,

$score(\mathrm{Re}\,l_i)$ is the score of individual relation which is obtained through training,

δ is set to 0.1 to avoid zero values.

The above formula is a variant of the ranking formula of local context query expansion except that we use relation path similarity instead of co-occurrence.

Besides the tokens, we also rank the path associated with each token Tk and select the top ranked path with start token Tk to the expanded path of Tk. The selection formula is shown as

$$path\_ex(Tk) = \{path(Tk,t,s) \mid path\_score(Tk,t,s) = \max_{\substack{t \in Q \\ s \in D_w}} \{path\_score(Tk,t,s\}\} \quad (3.1.5)$$

(3) We add top k tokens denoted as $\{Tk_1 ... Tk_k\}$ to the original query. We set the weight of original query terms to be 1.0 and the weight of ith expanded token to be (1-0.9*i/k). We use the expanded set to perform document and passage retrieval. We add $\{path\_ex(Tk_i) \mid 1 \le i \le k\}$ to the set of paths derived from the original question to perform dependency matching for answer extraction.

3.2 Training Individual Relation Weights

As explained in the previous section, the relevance of the expanded token Tk is judged by its relation paths linking to the tokens in the query. And each relation path is a sequence of individual relations. Under the assumption that each relation appears independent of the other relations in the same path we have

$$path\_score(Tk,t,s) = \prod_{\substack{Tk \in s \wedge t \in s \wedge \\ \mathrm{Re}\, l_i \in path(Tk,t)}} score(\mathrm{Re}\, l_i) \quad (3.2.1)$$

Therefore for each type of relation in the dependency parsing tree, we need to train $score(\mathrm{Re}\, l_i)$. We use TREC 8 and TREC 9, QA sentence pairs to perform training. We denote each QA pair as (Q_i,A_i). We then collect the top 50 snippets returned by Google for each question and perform sentence splitting and dependency parsing and select the relevant paths from the set of parsing trees of the snippets. A path p in the snippets of Qi denoted as (Start_Token, Rel_1... Rel_k...Rel_m, End_Token) is relevant if $Start\_Token \in A_i$ and $End\_Token \in Q_i$.

$$score(\mathrm{Re}\, l_i) = \log(C_{\mathrm{Re}\, l_i \in relevant\_path} + 1) / \max_{1 \le i \le N} \{\log(C_{\mathrm{Re}\, l_i \in relevant\_path} + 1)\} \quad (3.2.2)$$

Where

$C_{\mathrm{Re}\, l_i \in relevant\_path}$ is the number of $\mathrm{Re}\, l_i$ in relevant paths

N is the total number of relation types

According to the formula, the score of the relation is proportional to the probability that it is in a relevant path. In other words the more often a "good" expansion term is inferred by the relation, the higher the score it will get. We normalize the score to be between 0 and 1 by dividing the score by the maximum score.

3.3 Evaluation Results and Discussions

We perform document and passage retrieval using the query expansion technique described above. Figure 2 shows the precision-recall graph of the document ranking task.

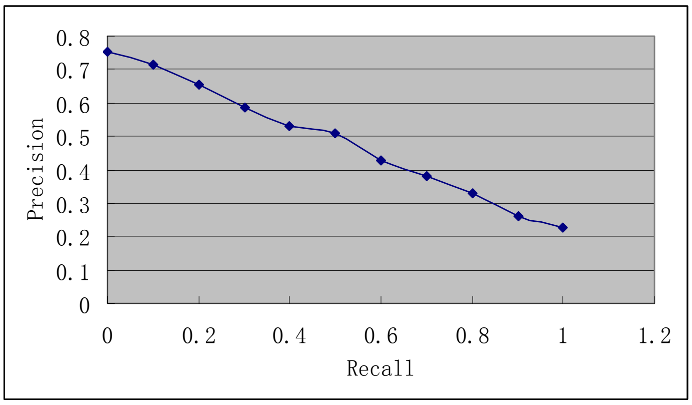

Figure. 2 Precision-recall graph of document ranking task (RUN NUSCHUAR1)

4 Semantic Answer Nugget Extraction on the Web

To facilitate the answering of topic-related factoid and list questions we use web resources as supporting documents to perform answer nugget extraction. We use http://www.answers.com/ as a search engine to get topic related web documents. And we perform semantic based answer nugget extraction on these documents. Finally, we project the answer nuggets back to corpus and verify the correctness of the answer nuggets.

4.1 Collecting and Pre-processing of Web Pages

For each topic, we input the topic as the query string to http://www.answers.com. The engine will return a resource page if the topic is in its database. Otherwise it will return the search result by Google. We use the resource page as our external web resource. We perform web page segmentation and classification on the resource page. For each resource page we classify the page segments into three classes, namely lists, descriptive data segments in natural language and commercial advertisements. We will remove all commercial segments as we will only use non-commercial segments to perform answer nugget extraction. Since the semantic answer selection works only on natural language sentences, we have to assign the surrounding texts as descriptive titles for tables and lists and use these titles to perform semantic matching with the questions.

4.2 Semantic Answer Extraction

Our goal is to match the semantic structure contained in the question with sentences from the web resources containing answer nuggets. After ranking the sentences we use the top k matched NEs (if the question has an NE typed answer target) or arguments (if the question has no NE typed answer target) as our candidate answer nuggets. There are two major challenges for this semantic matching. One is the different syntactic representations of the same semantic structure. The other is the use of different lexical terms to refer to the same concept. To tackle these problems, we use a shallow semantic parser to unify different syntactic representations into the same semantic representation, and we use WordNet and eXtended WordNet to find synonyms and semantically related verbs.

4.2.1 Shallow Semantic Parsing

To capture the semantic structures contained in a sentence, we need to identify verbs and their arguments. We also need to label the arguments with their semantic roles. This goal is achieved by performing shallow semantic parsing. The shallow semantic parser we use is the ASSERT parser, which is trained on the PropBank (Kingsbury *et al.,* 2002) corpus and uses support vector machine classifiers.

PropBank was manually annotated with verb-argument structures. Following the annotation rules in PropBank, the ASSERT parser tags the arguments of a verb with labels from ARG0 up to ARG5. Although such semantic roles are verb-specific, some labels such as ARG0 and ARG1 tend to be general to all verb classes. For example, for any transitive verb, ARG0 is always the subject, and ARG1 is always the direct object. Besides these core arguments, ASSERT also tags adjunctive arguments. Examples are ARGM-LOC for locatives and ARGM-TMP for temporal. Figure 3 shows a sample output of ASSERT containing the parsing result of the question and several answer candidate sentences.

Q: [ARG2-FROM Where] was [ARG1 the first Burger King restaurant] [TARGET opened]

P1: [ARG1 The first Burger King] [TARGET opens] [ARGM-LOC in Miami]

P2: [ARG0 Burger King Corp.] [TARGET announced] [ARGM-TMP Thursday] [ARG1 it has canceled the rights of an Israeli company to operate a controversial franchise in the West Bank and ordered the restaurant to remove the Burger King brand from the site immediately]

P2: Burger King Corp. announced Thursday [ARG0 it] has [TARGET cancelled] the rights of an Israeli company [ARGM-PNC to operate a controversial franchise in the West Bank] and ordered the restaurant to remove the Burger King brand from the site immediately

P2: Burger King Corp. announced Thursday it has canceled the rights of an Israeli company to [TARGET operate] [ARG1 a controversial franchise in the West Bank] and ordered the restaurant to remove the Burger King brand from the site immediately

Figure.3 Sample Output By ASSERT Parser

short) as a verb-argument structure obtained from a sentence by the ASSERT parser. A frame consists of a verb, which we call the predicate, and a set of arguments. The arguments include both core arguments and adjunctive arguments. Each argument is associated with a label such as ARG0 and ARG1 to indicate the semantic role of the argument. Therefore, a frame *F* can be represented as *F=(v,A)*, where *v* is the predicate and *A* is the set of arguments. Each element *a* in *A* is a pair consisting the argument label and the argument text, represented by *a=(l, T)*, where *l* is the label and *T* is the set of terms that the argument contains. Because a sentence may contain more than one semantic structure, a sentence is represented by a set of frames.

4.2.2 Verb Expansion using WordNet

Before we show our similarity scoring function, we first look at verb similarity scores. An answer passage can express the same semantic structure of the question with a different verb that is either of the same meaning as the verb in the question or semantically related to the verb in the question. Therefore, when matching semantic frames, we need to consider the semantic similarity between two verbs. We use WordNet and eXtended WordNet to measure this verb similarity.

Our verb similarity function is very similar to the weighting function in Moldovan *et al.,* 2002. Suppose we want to measure the similarity between two verbs *v1* and *v2*. We start from one of the verbs, say, *v1*. This original verb is assigned a score of 1. We select the synset that corresponds to the first sense of *v1* in WordNet. All words in this synset get the same score as the original word. From this synset, we follow the links to other synsets with relations such as hyponyms and entailment. We also follow the gloss links and reverse gloss links provided by the eXtended WordNet. A gloss link from synset *S1* to synset *S2* means *S2* appears in the gloss of *S2*, and a reverse gloss link from synset *S1* to synset *S2* means *S1* appears in the gloss of *S2*.

If from *v1* we follow the relation *R* to a synset which contains the word w, then the score for the word w is $1 \times W_R$, where W_R is the weight for relation *R*. If we follow the link further from *w* to another synset which contains the word *u* by relation *S*, then the score of the word *u* is $1 \times W_R \times W_s$. Such expansion continues until we reach a certain depth. In our experiments, we set the depth to 2 because our preliminary experiments show that a deeper expansion does not improve the performance. The weights of the relations are the same as that used in Moldovan *et al.,* 2002. We also penalize synsets that are more commonly used. Each synset gets a generality score *G* which is defined as

$$G_s = \frac{C}{C + N_{r-gloss}} \quad (4.2.2.1)$$

where *C* is a constant, and $N_{r-gloss}$ is the number of glosses in which the member of synset *S* appears. We set *C*=500. After taking into account this penalizing weight, the score of the word w is therefore $s_v \times R_{v,w} \times Gs_w$, where s_v is the score of the previous word *v*, $R_{v,w}$ is the relation connecting *v* and w, and s_w is the synset containing w.

After we expand the verb *v1*, we check if *v2* appears in the expanded set. If it does, then it is assigned the score as explained above. If not, *v2* is assigned a score of 0. We denote this similarity score between *v1* and *v2* as $Sim_V(v_1, v_2)$.

4.2.3 Semantic Matching

First, we define the similarity scores between two frames. Let *F1= (v1, A1), F2 = (v2, A2)*. We divide the similarity score into two components, one indicating the similarity between the verbs, and the other indicating the similarity between the arguments.

$$Sim(F_1, F_2) = \alpha \times Sim_V(v_1, v_2) + (1-\alpha) \times Sim_A(A_1, A_2) \quad (4.2.3.1)$$

where $Sim_A(A_1, A_2)$ denotes the similarity score between two argument sets, and α is a weighting parameter that can be tuned. Our experimental results show that α does not affect the performance much within a certain range. Therefore we fix α to be 0.5.

The similarity between the two sets of arguments is measured at the lexical level. We do not use WordNet to expand the terms in the arguments because many of the arguments are named entities such as persons and organizations, for which finding similar terms is not so meaningful.

To precisely match the argument sets of two frames, we should do pairwise matching of the arguments, that is, matching ARG0 in the first frame with ARG0 in the second frame, and ARG1 in the first frame with ARG1 in the second frame, etc. However, we choose to do a fuzzy matching by considering all arguments in a frame together as a bag of independent terms. There are two reasons for doing fuzzy matching: (1) ASSERT often makes mistakes and therefore does not tag the arguments consistently, especially for adjunctive arguments, and (2) since we consider semantically related verbs, the semantic roles of the arguments may be different in different frames. Our preliminary experimental results also show that considering all arguments together is better than considering them separately.

We use Jaccard coefficient to measure the similarity between two sets of arguments. Suppose we are to compute $Sim_A(A_1, A_2)$, where *A1* and *A2* are two argument sets:

$$A_1 = \{(l_{1,1}, T_{1,1}), (l_{1,2}, T_{1,2}), ..., (l_{1,m}, T_{1,m})\} \quad (4.3.2.2)$$

$$A_2 = \{(l_{2,1}, T_{2,1}), (l_{2,2}, T_{2,2}), ..., (l_{1,n}, T_{1,n})\} \quad (4.2.3.3)$$

$l_{i,j}$ is the argument label of the *j*th argument of *Ai*, and $T_{i,j}$ is the set of terms in the *j*th argument of *Ai*. Let

$$T_1 = \bigcup_{i=1}^{m} T_{1,i} \text{ and } T_2 = \bigcup_{i=1}^{n} T_{2,i} \quad (4.2.3.4)$$

We then remove the stop words from *T1* and *T2*. Let the sets of terms after stop word removal be *T1'* and *T2'*. We then define the similarity between *A1* and *A2* as

$$Sim_A(A_1, A_2) = \frac{|T_1' \bigcap T_2'|}{|T_1' \bigcup T_2'|} \quad (4.2.3.5)$$

Both the question and the answer passages may contain more than one semantic frame. We compute pairwise frame similarity scores between the question and an answer passage, and pick the maximum score as the semantic similarity between the question and the answer passage.

Finally, we use the semantic similarity scores to rank passages. For a question, we first use a density based passage retrieval method to retrieve the top 100 passages. We than rank these 100 passages based on their semantic similarities to the question, as defined above.

4.3 Answer Projection and Verification

After we obtain the answer nuggets from the web, we need to project the answer nuggets back to the corpus. For list questions, we focus more on the recall so we only perform answer projection on the document level. For factoid questions after the projection step we use dependency relation based answer ranking to verify if the answer nugget from the web is correct in the local context.

4.4 Evaluation Results

Table. 1 shows the evaluation result for TREC-14 factoid and list questions based on run NUSCHUA1.

Table 1. Performance for factoid and list questions

	NUSCHUA1	TREC Highest	TREC Medium
Accuracy for factoid questions	0.666	0.713	0.152
Accuracy for list questions	0.331	0.468	0.053

We find that our performance for factoid questions is improved over our result last year. The main reason is that after query expansion our system is more capable of handling short questions. However, our semantic answer nugget selection does not perform very well on list questions. The major problem is due to the recall of the semantic parser and the strict matching criteria we imposed on the frame matching, which will reduce the recall of finding answers for list question.

5 Definition Generation for Topics

We consider it important to identify precise and complete definition sentence for topics because it facilitates the answering of factoid and list questions, and more importantly, it helps to answer the Other questions. In TREC-13, we applied a soft pattern model to boost the recall of definition sentence retrieval. This year, we use the improved bigram soft pattern model (Cui *et al.*, 2005), and combine it together with external knowledge, to identify precise definition sentences. Note that for external knowledge, we choose to utilize specific websites rather than general search engines, so that we get more precise results.

5.1 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 http://www.answers.com/. The definitions are downloaded through pre-written wrappers for the website.

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:

$$Weight_{centroid}(w) = \frac{\log(Co(w, sch\_term) + 1)}{\log(sf(w) + 1) + \log(sf(sch\_term) + 1)} \times idf(w) \qquad (5.1.1)$$

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*)[4], as a measure 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.

5.2 Generic Soft Pattern Model

After performing the statistical ranking step above, we have a list of ranked sentences with definition sentences ranked highly. However, not all sentences that are ranked highly are definition sentences, although all of these sentences are related to the topic.

In most TREC QA systems, definition patterns are manually constructed in a labour intensive manner, and are usually in the form of regular expressions. Such patterns require exact matching and hence we call them *hard patterns*. We observe that definition sentences such as "… the weed kudzu, a vine planted for soil stabilization that has grown like wild ..." follows certain patterns, but often with minor variations for a variety in writing styles. Hard patterns usually fails in matching such linguistic variations in vocabulary and syntax, and learned hard patterns cannot match definition sentences that are not seen in the training data. Therefore we propose and employ a soft pattern model discussed in (Cui *et al.*, 2004), and further improved it in (Cui *et al.,* 2005) to produce a theoretically sound generic soft pattern model. In (Cui *et al.,* 2005), a bigram soft pattern model and a profile HMM soft pattern model are proposed, and we apply the bigram soft pattern model here because the profile HMM model requires more training instances to converge.

For a definition pattern containing the search target, we consider the tokens on the left of the search target, *left_seq*, separately from those on the right of the search target, *right_seq*, and compute their respective scores separately. We combine the two scores using linear interpolation:

$$\begin{aligned} \text{score}\left(left\_seq, right\_seq\right) &= \alpha\,\text{score}\left(left\_seq \mid \mu_{left}\right) \\ &+ \left(1-\alpha\right)\text{score}\left(right\_seq \mid \mu_{right}\right) \end{aligned} \qquad (5.2.1)$$

where μ_{left} and μ_{right} are the bigram soft pattern models for the left and right sequences respectively.

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

In the original bigram soft pattern model used in TREC-13, the probability of a sequence is computed simply as a product of probabilities of bigrams. For our improved bigram soft pattern model, we apply linear interpolation of unigrams and bigrams to represent the probability of bigrams to smooth the distribution to generate more accurate statistics for unseen data, as well as to incorporate the conditional probability of individual tokens appearing in specific slots. For a given bigram model μ with slots S_1 to S_L, a sequence of pattern tokens t_1 to t_L is modeled as follows:

$$\text{score}(t_1,\ldots,t_L \mid \mu) = \frac{1}{L}\left(\log P(t_1 \mid S_1) + \sum_{i=2}^{L} \log\left(\lambda P(t_i \mid t_{i-1}) + (1-\lambda) P(t_i \mid S_i)\right)\right) \tag{5.2.2}$$

where $P(t_i \mid S_i)$ is the conditional probability of token t_i appearing in slot S_i. The unigram and bigram probabilities are estimated using maximum likelihood estimation, and Laplacian smoothing is applied to these probabilities. For more details on our bigram soft pattern model, please refer to (Cui *et al.*, 2005).

5.3 Manually Constructed Patterns

On top of centroid-based weighting and soft pattern matching, we also optionally use a set of manually constructed patterns for matching definition sentences. This set of patterns includes the subset of patterns we used for TREC-12 that are used in TREC-13. The set consists mainly consisting of appositives and copulas patterns, which are high-precision patterns represented in regular expressions, such as "<SEARCH_TERM> is DT$ NNP". For this year's TREC QA, we used additional high-precision patterns used to match numeric data, such as years and distance units.

Such hard matching patterns are used on top of the soft patterns because a small number of good definition sentences are dropped due to the imposed cut-off of ranking scores by centroid-based weighting and soft pattern matching, and we want to capture these sentences. Also, sentences containing numerical data are presented in a large number of formats and are not given very high scores by the soft pattern models.

Hence, the system works by first ranking all the sentences using centroid-based ranking and soft pattern matching, and then taking the top ranked sentences as candidate definition sentences. Optionally, for boosting the recall of the definition sentences, it then examines those lower ranked sentences which are not included in the candidate definition sentences, and adds those sentences matched by any of the manually constructed patterns into the list of definition sentences.

5.4 Redundancy Removal

Like in TREC-13, this year's TREC QA guidelines requires that systems remove nuggets that are already covered in the topic-related factoid and list questions from their list of definition nuggets. Our system performs a two-stage redundancy check when selecting definition sentences into the final answer. First, we define the list of sentences used to answer the factoid and list questions for the same topic as *factoid sentences*. For selecting *N* sentences for the final answer, we apply the following selection process on our ranked list of sentences:

1. Add the first sentence in the ranked list of sentences into the list of answer sentences.
2. Let the next sentence in the ranked list of sentences be *next_stc*.
    a. If there is a factoid sentence *factoid_stc* such that *sim*(*next_stc*, *factoid_stc*) ≥ 0.85, skip to step (3).
    b. If there is already a selected answer sentence *answer_stc* such that *sim*(*next_stc*, *answer_stc*) ≥ 0.70, skip to step (3).
    c. Otherwise, we add *next_stc* to the list of answer sentences.
3. If the list of answer sentences already has *N* sentences, we terminate the process. Otherwise, go back to step 2.

Here, we measure the similarity between two sentences using simple cosine similarity with each term weighed by its inverse document frequency (IDF). Since the answers to factoid or list questions tend to account for very small fraction of the sentences, we apply a stricter similarity threshold on these sentences.

For this year, we choose to return only full sentences as our definition nuggets, without attempting to extract the relevant substrings of the sentences. This is unlike TREC-13, where heuristic rules were used to extract only the relevant parts, such as the appositive part. The reason is that the context of the sentence is often lost when such extraction is done.

5.5 Evaluation Results

This year, we submitted three runs for the Other questions. The first run produces 14 definition sentences using only soft pattern matching. The second run produces 12 definition sentences using soft pattern matching and 2 using the hard pattern matching rules as used in TREC-13, but not including the numeric ones. The third run produces 12 definition sentences using soft pattern matching and 6 using the hard pattern matching rules, including the numeric ones. The average F_3-scores are shown in Table 2.

Table 2. Performance for Other questions

	NUSCHUA1	NUSCHUA2	NUSCHUA3
Avg F_3-score	0.195	0.193	0.211

From the scores, we see that our bigram soft pattern model performs as well as the manual non-numeric hard pattern matching rules, which already has a performance that is much better than the median average F_3-score of 0.156 across all the 71 runs. However, since number matching is a hard problem in general, adding some hard numeric matching rules to augment the soft pattern model in our third run actually boosts the results, even though the returned answers are much longer now.

This year, the topics are extra challenging for answering the Other questions, reflected by the across-the-board low scores, because a large number of the topics are events. A reason is because events can span across a long period of time and involve a large number of entities. At the same time, having to exclude all definition nuggets that have been covered by the topic-related factoid and list questions continue to make answering the Other questions difficult. We continue to observe that most of the important aspects of a topic have already been asked in the factoid and list questions, leaving little else for answering the Other question.

6 Conclusion

We have reviewed the newly-adopted techniques in our QA system. They include using dependency relation analysis for query expansion, using semantic relation analysis for answer nugget extraction from the web and using bigram soft pattern model for definition sentence selection. While these techniques have improved our previous QA system, we note that more improvements may be pursued in future work. First, the recall of semantic parsing is not high enough to cover most of the questions. Therefore to increase the recall of semantic parsing will be one of our future works in applying semantic relation analysis to QA. Secondly, more experiments should be conducted to figure out the effect of query expansion on the specific type of questions, in particular for questions with different lengths. Third, further work needs to be done for answering Other questions for events. Also, there is a need to find ways to integrate numberic matching into the soft pattern models.

References

[Cui *et al.,* 2004] H. Cui, K. Li, R. Sun, T.-S. Chua and M.-Y. Kan, *National University of Singapore at the TREC-13 Question Answering Main Task*, Proceedings of the 13th Text Retrieval Conference (TREC 2004).

[Cui *et.al.,* 2005] H. Cui, M.-Y. Kan and T.-S. Chua, *Generic soft pattern models for definitional question answering,* Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval, Salvador, Brazil , Aug 15-19, pp. 384 - 391.

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

[Pradhan *et al.,* 2004] S. Pradhan, W. Ward, K. Hacioglu, J. H. Martin and D. Jurafsky, *Shallow Semantic Parsing using Support Vector Machines*, Proceedings of HLT/NAACL '04, Boston, MA, 2004.

[Xu *et al.*,1996] J. Xu, W. Bruce Crof, *Query expansion using local and global document analysis,* Annual ACM Conference on Research and Development in Information Retrieval Proceedings of the 19th annual international ACM SIGIR conference on Research and development in information retrieval Zurich, Switzerland,1996, pp. 4-11.

[Kingsbury *et al.,* 2002] P. Kingsbury, M. Palmer and M. Marcus, *Adding Semantic Annotation to the Penn TreeBank,* Proceeding of the Human Language Technology Conference, San Diego, California, 2002.

[Moldovan *et al.,* 2002] D. Moldovan and A. Novischi, *Lexical Chains for Question Answering,* Proceeding of COLING 2002, pp. 674-680

[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.