UNT at TREC 2004: Question Answering Combining Multiple Evidences

Jiangping Chen, He Ge, Yan Wu, Shikun Jiang

School of Library and Information Sciences
University of North Texas
P.O. Box 311068, Denton, TX 76203
{jpchen, hg0022, ywu0015, sj0071}@unt.edu

1 Introduction

Question Answering (QA) aims at identifying answers to users' natural language questions. A QA system can release the users from digesting large amount of text in order to locate particular facts or numbers. The research has drawn great attention from several disciplines such as information retrieval, information extraction, natural language processing, and artificial intelligence. TREC QA track has provided comparable QA system evaluation on a set of test questions since 1999. The degree of difficulty of the test questions has increased substantially in recent two years, which push the research toward applying more sophisticated strategies and better understanding of English texts.

Question answering is very challenging due to the ambiguity of the questions, complexity of linguistic phenomena involved in the documents, and the difficulty to understand natural languages. More challenging is to locate short snippets or answers from a document collection with texts written in different languages, which is within our research interests focusing on cross-lingual or multilingual information access and retrieval. We have decided to participate in TREC 2004 Question Answering Track as our first step toward exploring advanced multilingual information retrieval. Our goal of this year is to develop a prototype automatic question answering system that can be continually expanded and improved.

Our prototype QA system, named EagleQA, made use of available NLP (Natural Language Processing) tools and knowledge resources for question understanding and answer finding. This paper describes the overall structure of the system, NLP tools and lexical resources employed, our QA methodology for TREC 2004, QA test results & analysis, and our plan for future research.

2 System Overview

Current EagleQA system is comprised of 6 major subsystems: Question Processing, Document Annotation, Sentence Retrieval, Web QA, Answer Finding, and Answer Formulation. Following will briefly discuss each of the subsystems.

2.1 Question Processing

Question Processing subsystem accepts users' questions and performs several processing including linguistic analysis, keyword identification and expansion, and answer type identification. Linguistic analysis performs part-of-speech tagging and phrase bracketing on original questions. Keyword identification and expansion first extracts important words or phrases from the annotated question. A word or a phrase is regarded as important if it's not included in the stopword list of the system. The stopword list was generated manually by identifying words that occur frequently in previous TREC questions. Next, for each extracted noun and verb, its synonyms and derivation forms were identified based on WordNet 2.0 (www.princeton.edu). Those terms are added to the keyword list. Answer type identification is another important procedure in Query Processing. We developed a simple ontology for QA purpose. 16 top level categories were identified from previous TREC questions. Sample categories include ANIMAL, CODE, CURRENCY, LOCATION, NUMBER, ORGANIZATION, and PERSON.

2.2 Document Annotation

This year we didn't carry out our own IR experiments to find relevant documents for test questions. Instead, we used the ranked document list provided by NIST. We plan to use Lemur (http://www-2.cs.cmu.edu/~lemur/) as the search engine for QA in the future.

Our Document Annotation subsystem combines document annotation results from two NLP systems: LingPipe (http://www.alias-i.com/lingpipe/) and Minipar (Lin, 1994). LingPipe is used first to detect sentence boundaries, the identified sentences are sent to Minipar for part-of-speech tagging and named entity categorization. LingPipe can also perform named entity categorization and co-reference annotation. At last, we integrate the results of annotations from the two systems using an XML format. Figure 1 shows an example of the combined annotated text. The categorization results from LingPipe are identified after '*ling_type =*' in the xml brackets, while those from Minipar are labeled after '*mini_cat=*'.

```
<sent id="19">
  <TOK id="1" pos="U">Last</TOK>
  <TOK id="2" pos="N">week</TOK>
  <TOK id="3" pos="U">,</TOK>
  <TOK id="4" pos="DET">the</TOK>
  <TOK id="5" pos="N">literature</TOK>
  <TOK id="6" pos="N">prize</TOK>
  <TOK id="7" pos="V" base="go" subj="prize:TOK_6">went</TOK>
  <TOK id="8" pos="PREP">to</TOK>
  <TOK id="9" pos="N" mini_cat="LANG">Portuguese</TOK>
  <TOK id="10" pos="N">novelist</TOK>
<NP id ="1" mini_cat="PERSON" ling_type="PERSON">
  <TOK id="11" pos="U">Jose</TOK>
  <TOK id="12" pos="N" mini_cat="PERSON">Saramago</TOK>
</NP>
 <TOK id="13" pos="U">.</TOK>
</sent>
```

Figure 1: Text Annotation Using LingPipe and Minipar

2.3 Sentence Retrieval

The Sentence Retrieval subsystem identifies a certain number of non-duplicate sentences (500 sentences maximum for this year) from the annotated documents as sentence candidates which may contain an answer to each test question. The keyword lists and answer type information obtained in Question Processing are utilized to find matched sentences for each factoid and list question. For questions labeled other, the sentence retrieval subsystem returns the sentences that match the target as answer candidates.

2.4 Web QA

The Internet is a huge and unique knowledge base. Our Web QA subsystem attempts to make use of this knowledge resource by submitting the original test questions to *Google*. The short summaries returned by Google are annotated and analyzed. A list of answer candidates that match the answer type of each question is then identified. Their frequency information is also kept as a factor for ranking the candidate by the Answer Finding subsystem.

2.5 Answer Finding

Answer Finding subsystem applies multiple evidences to find answers for factoid and list test questions. Factors that are taken into account when ranking an answer candidate include: 1) answer type; 2) weight of the sentence; 3) distance to keywords in the same sentence; and 4) whether it is a candidate returned by Web QA

2.6 Answer formulation

Finally, the system combines the answers for different types of questions such as factoid, list, and other questions. The duplicate answers are filtered out from the list of answers to other questions. An answer file is formulated at the end of this stage for submission.

Figure 2 outlined the current architecture of our EagleQA system for TREC 2004.

3 NLP tools and Knowledge Resources

As mentioned in the introduction, we chose to make use of freely available NLP tools and knowledge resource and to integrate them into our QA system. Following describes the NLP tools and knowledge resource employed by the different subsystems of EagleQA to process test questions or documents.

3.1 LingPipe (http://www.alias-i.com/lingpipe/)

LingPipe, an open source NLP software, is developed by Alias-I, Incorporated (http://www.alias-i.com/). LingPipe is regarded as "a suite of Java tools designed to perform linguistic analysis on natural language data." LingPipe provides linguistic analysis functions such as sentence boundary detection, named entity detection for person, organization, and location, and within-document co-reference resolution. Evaluation of LingPipe system performance for various tasks can be found at http://www.alias-i.com/lingpipe/benchmarks.html.

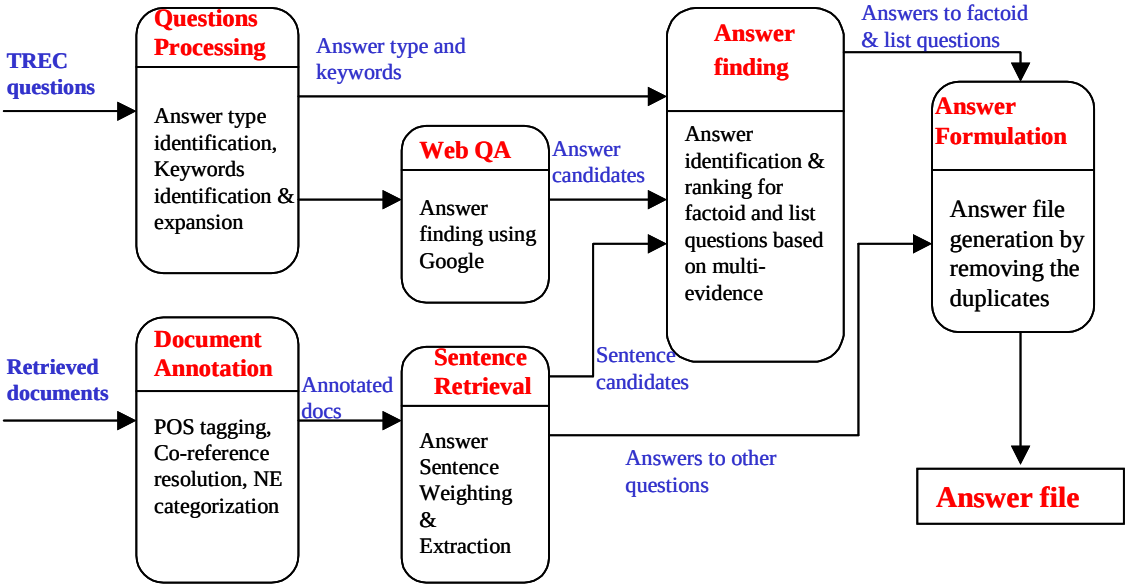

Figure 2. EagleQA Architecture

3.2 Minipar (http://www.cs.ualberta.ca/~lindek/minipar.htm)
MINIPAR, an efficient parser for English developed by Dr. Dekang Lin (1994), provides NLP functions such as part-of-speech tagging, phrase bracketing, and named entity categorization. An executive version can be downloaded from Dr. Lin's website. Minipar was used by our system in combination with LingPipe and WordNet to annotate questions and documents.

3.3 WordNet (http://www.cogsci.princeton.edu/~wn/) and Related Tools
WordNet (Miller, 1990) is the well-known English ontology freely available on the Web and covers the vast majority of nouns, verbs, adjectives, and adverbs from the English language. It has been widely used in many NLP applications and other QA systems (Harabagiu, et al., 2003).

In addition to the WordNet database itself, we made use of a Perl interface to WordNet: WordNet::QueryData developed by Jason Rennie (http://people.csail.mit.edu/u/j/jrennie/public_html/WordNet/). It allows the user direct access to the full WordNet semantic lexicon. For example, you can query synonyms, hyponyms, the gloss of a particular sense, and derived forms of a word.

3.4 Google (www.google.com)
As many other QA systems, we used Google.com to search for answers on the Internet. The Web QA subsystem submits the original questions to Google for short summaries from which we identify a list of possible answer candidates. The test results showed that

our approach was too naïve and the performance was not satisfactory. Further analysis will be conducted to improve the strategies of identifying and ranking answer candidates from the search results.

4 QA Methodology for 2004

We basically employed similar strategy to factoid questions and list questions except that a threshold was applied to list questions in order to determine the number of answers that should be returned. The threshold was predetermined based on experiments using last year's test questions. Other questions are quite different from factoid questions. Therefore we used a different approach to answer them.

4.1 factoid questions and list questions

As depicted by figure 2, factoid questions and list questions were handled following the procedures of question processing, document annotation, sentence retrieval, Web QA, answer finding, and answer formulation. Following we will describe more specifically approaches for some of the subsystems, such as question processing, sentence retrieval, and answer finding.

For question processing, EagleQA first performed a shallow co-reference resolution to replace pronouns in questions with the actual targets. For example, in question 4.5 "*Which was the first movie that he was in?*" "*he*" was replaced by the target "*James Dean*". Then we used Minipar to tag each question and WordNet to expand nouns and verbs in the question, as described in 2.1. Final result for each question included the answer type and a list of keywords with expansions if any. For example,

Question: *Which was the first movie that James Dean was in?*
Answer type: movie
Keywords: *movie(synonym: movie, film, picture, moving picture, moving-picture show, motion picture, motion-picture show, picture show, pic, flick) / James Dean /*

After annotating the retrieved documents using LingPipe and Minipar, we employed a simple strategy to find sentences that may contain the answers. Two factors were considered for ranking the sentences: number of question keywords occurred in the sentence, and whether the sentence contains the same answer type as the question. Higher weights were assigned to sentences that contain more keywords and named entities that are annotated the same as the answer type of the question.

At the answer finding stage, we first attempted to identify possible answer candidates. Named entities annotated by LingPipe and Minipar were identified as answer candidates if their categories are the same as those of the question. However, the document annotation using LingPipe and Minipar can only identify named entities in certain categories, such as *person*, *country*, *location*, *city*, *money*, *number*, and *organization*. Many questions ask about names in other categories, or cannot be answered using entity-based strategy. We therefore developed following additional strategies for answer candidate identification:

- WordNet hypernyms. For those questions asking about certain categories such as *animal*, *disease*, *plant* and *color*, we evaluated each noun or noun phrase in the sentence using knowledge about hypernym relationship in WordNet. For example, question "*What is their gang color?*" the answer type is *color*. For each noun in a candidate sentence, we used WordNet to check whether 'color' is one of its hypernyms. If conformed, the word or phrase was regarded as an answer candidate.
- Name lists. Some frequently asked questions, such as "*Who was the 23rd president of the United States?*" can be easily answered if a list of US presidents has been stored in the system. We therefore manually developed about thirty short lists to store names about US presidents, NBA teams, candy brands, state nicknames, and newspaper agencies. If a noun or noun phrase matched one of the names in an appropriate list, it was regarded as an answer candidate. The criterion to choose the appropriate list depended on the degree of match between keywords of the question and the title of the list.
- Additional named entity patterns. In order to compensate for the incomplete document annotation, we developed a pattern set to identify names or titles of a book, a move, a poem, and a song. Also included are patterns to identify terms or phrases regarding speed, temperature, and age.
- Answer patterns. Some questions are difficult to apply entity-based strategy. For instance, question "*How did James Dean die?*" is ambiguous with regard to the answer type. We therefore developed simple patterns for answering such questions. These patterns were derived from the answers to the similar test questions of 2003.

Once the answer candidates were identified, the next step was to rank them in order to determine the final answer. The ranking strategy depended on following three evidences:

- The weighting score of the answer sentence obtained in Sentence Retrieval subsystem. This score is determined by two factors: number of question keywords and answer type presence if applicable, as discussed above.
- The weighting score of the answer candidates. Candidates obtained through different strategies outlined above might be assigned a different weighting score. For example, answer candidates obtained from document annotation had a higher score than those from additional named entity patterns.
- Web QA. If an answer candidate was also returned by the Web QA subsystem, it received a higher ranking.

A formula considering the above factors was used to calculate the final score for each candidate. The parameters in the formula were trained using a small number of test questions of last year.

4.2 Other questions

The other questions were previous definition questions. Other questions require QA systems to provide information about the targets in addition to answers to the factoid and list questions for the same targets. The usual ways to answer definition questions include developing heuristic linguistic patterns to find matched sentence segments (Hildebrandt, Katz, & Lin, 2004). Due to the time constraints, we were unable to develop training material and heuristic patterns for other questions. We instead employed a simple

strategy: just return the whole sentences which include more than 50% of the words in the targets. In ranking candidate sentences, higher scores were given to sentences which include the whole targets as phrases. This simple strategy sacrifices precision based on current performance measures.

5 Results & Analysis

We submitted three runs, namely UNTQA04M1, UNTQA04M2, and UNTQA04M3. Their scores are listed in Table 1. The three runs applied the same program for other questions. The difference between UNTQA04M1 and UNTQA04M2 was that UNTQA04M2 took into account Web QA or Google results for weighting the answer candidates, but UNTQA04M1 didn't. The difference between UNTQA04M1 and UNTQA04M3 was the different post processing for other questions. UNTQA04M1 returned non-duplicate answers for each other question. Non-duplicate means if an answer sentence had returned an answer for factoid or list questions for the same target, it was removed from the answer list for the other question. UNTQA04M3 didn't perform this process of removing duplicate answers. The results showed that the process produced no effect on system performance.

Table 1. Official Results

Run	Factoid (Accuracy)	List (Average F)	Other (Average F)	Final Score
UNTQA04M1	0.187	0.128	0.305	0.202
UNTQA04M2	0.196	0.123	0.305	0.205
UNTQA04M3	0.187	0.127	0.307	0.202
median	*0.170*	*0.094*	*0.184*	

Run UNTQA04M2 did slightly better than the other two runs, mainly because of the use of Google results. Event though the three runs were all above the median, there is big room for further improvement.

We are still in the process of analyzing our QA results. The analysis aims at discovering the weakest point of the system so that we can take it as the start point for future development.

6 Future Research

Our QA system is still at a very early development stage. Some of the subsystems had not been fully tested before the TREC experiments due to time constraints. We will continue our effort to develop and evaluate the system.

We plan to use Lemur (http://www-2.cs.cmu.edu/~lemur/) as our search engine for document retrieval for QA. Based on our testing using 2003 questions, Lemur could retrieve more relevant documents than NIST search engine.

For question processing and answer finding, we are attempting template-based techniques in combination with machine learning to improve the system performance. Preparing training material will be our first step of exploring these new strategies. We expect to improve the QA performance of our EagleQA system in the near future.

7 References

Harabagiu, S., Moldovan, D., Clark, C., Bowden, M., Williams, J., & Bensley, J. (2003). Answer Mining by combining extraction techniques with abductive reasoning. In *Proceedings of The Twelfth Text Retrieval Conference (TREC 2003).*

Hildebrandt, W., Katz, B., & Lin, J. (2004). Answering definition questions with multiple knowledge sources. *Proceedings of the 2004 Human Language Technology Conference and the North American Chapter of the Association for Computational Linguistics Annual Meeting (HLT/NAACL 2004),* May 2004, Boston, Massachusetts.

Lin, D. (1994). PRINCIPAR---An Efficient, broad-coverage, principle-based parser. In *Proceedings of COLING-94.* pp.42--488, Kyoto, Japan.

Miller, G. (1990). WordNet: an on-line lexical database. *International Journal of Lexicography*, 2(4), Special issue.