Parsing tree matching based question answering

Ping Chen
Dept. of Computer and Math Sciences
University of Houston-Downtown
chenp@uhd.edu

Wei Ding
Dept. of Computer Science
University of Massachusetts –Boston
ding@cs.umb.edu

Timothy Simmons
Carlos Lacayo
Dept. of Computer and Math Sciences
University of Houston-Downtown

Abstract

This paper describes the Question and Answering system participating Question Answering track in Text Analysis Conference organized by National Institute of Standard and Technology 2008. Our Question and Answering system attempts to use a human style of logic to search their respective document sources and return possible answers to a question.

1. Introduction

The Question and Answering Track System was developed to meet the requirements of the Text Analysis Conference (TAC) 2008 Question Answering (QA) Track [5]. The system attempts to provide rigid and squishy list answers to questions provided by the TAC QA Track by searching the BLOG06 corpus provided by the University of Glasgow [1]. The BLOG06 corpus is a collection of BLOG sites that consists] of text entries by BLOG authors and corresponding replies by subscriber or visitors. Due to time and budget constraints, we only used the 50 documents for each question provided by NIST.

2. Question Answering System Architecture

Figure 1 shows the architecture of our Question Answering system. The system includes two parts: indexing component and question answering component.

2.1. Indexing component

The goal of indexing component is to build a word-based index based on the source documents. When we search for information related to a word, the sentences containing this word will be located and extracted directly, which is more efficient than a string matching for every keyword in every question. This component includes the following steps:

1. **Cleaning**
   The cleaning step is responsible for stripping tags from HTML files and converting HTML codes back to their text equivalents. For example, the program would convert '&quot' in a text string back to its double quote text equivalent of '"' or '&amp' to '&'.
2. **Sentence segmentation**
   We developed a simple sentence segmentation program, which separates sentences based on punctuation (e.g., ?, !, .).

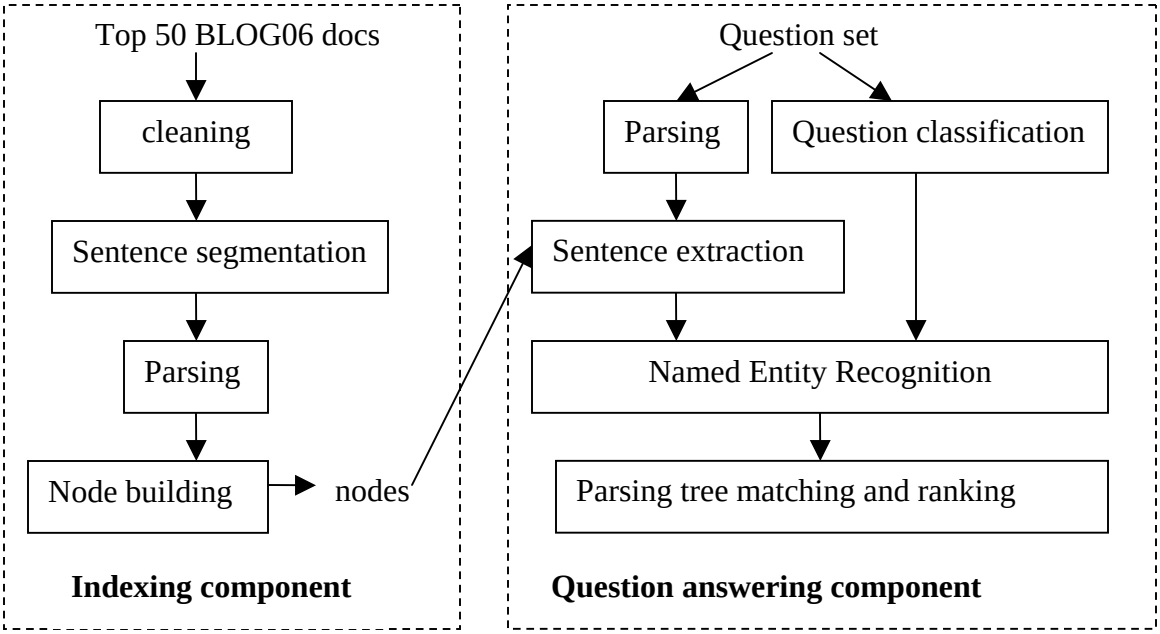

Figure 1 Question Answering System Architecture

3. Parsing

After cleaning and sentence segmentation, sentences are parsed with a dependency parser, Minipar. Dependency parsers have been widely used in information extraction. An evaluation with the SUSANNE corpus shows that Minipar achieves 89% precision with respect to dependency relationships [5].

4. Nodes building

After parsing, dependency relations from different sentences are integrated. The integration process is straightforward. One dependency relation includes two words/nodes, one head word/node and one dependent word/node. Nodes from different dependency relations are merged into one as long as they represent the same word. An example is shown in Figure **2**, which merges the following two sentences:

"Computer programmer writes software."

"Software is a useful tool."

After merging, we save each node, its dependent nodes, and the sentence id into a file. Hence, each node file contains all occurrences of a word and the locations of sentences that contain it. Node files will be used in the question answering component to retrieve relevant sentences and generate answers.

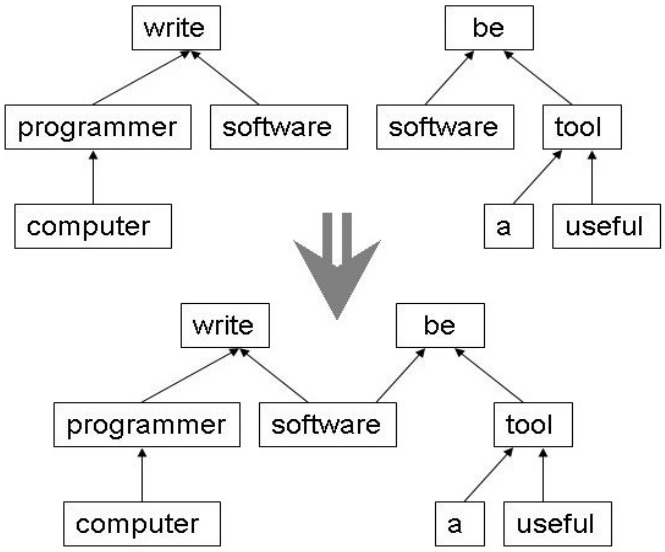

Figure 2 Integration of two parsed sentences

2.2. Question Answering Component

The main program Matcher.exe is responsible for processing the questions, searching for matching sentences, parsing the matched sentences, calculating the scores for each of the sentences, and outputting the answers to a results file. The goal of question answering component is to generate answers based on questions and source documents. It consists of the following steps:

1. **Question classification**

   We built a rule-based question classification system, and initial testing using QA track 2003-2007 data achieves 80%- 90% accuracy. Here are the question, categories, which is similar as proposed in [4]:

   animal, city, code, color, count, country, creative, date, description, distance, entity, event, food, group, individual, instrument, language, location, money, order, organization, percent, period, product, reason, religion, size, speed, sports, state, symbol, technique, temp, term, title, vehicle, weight

   After testing with QA track 2008 data, we found that these categories cannot describe the 2008 data well, which hurt the performance of our system.

2. **Relevant sentence extraction**

   Every question is parsed with Minipar. Nouns, verbs, adjectives, and adverbs are extracted and compared with the nodes built with indexing component. If any nodes match, the corresponding sentences are loaded and become the candidate sentences.

   This step has evolved through much iteration to fit the purpose of the QA Track 2008. Initially the BOOST libraries [5] were used to perform regular expression searching on the entire BLOG06 collection to

selectively choose documents that only contained key words or the target. To accomplish this, the dictionary was used to improve document retrieval by allowing for "stem words" to be included in the search. For example, a search on the word "sleep" should also return documents with "slept", "sleeping", or "sleeps". Due to the composition of the BLOG06 corpus, this approach proved to be inefficient due to the collection size and the minimal amount of text per file. To further refine searching a tagging method was used to identify only nouns in the target. The tagger from the Tsujii Laboratory would read a file containing the current question and produce a tagged sentence [6]. QA system would then read the output file and set the target to only the nouns specified by tagger.

3. Named Entity Recognition

Named Entity Recognition is performed with GATE to identify when a candidate sentence contains the type of information requested in a question. GATE is a Natural Language Processing system written in JAVA [3]. It ANNIE component analyzes the sentences and marks specific areas of each sentence such as "location" or "organization". GATE will export the results to an HTML file. The HTML files identify regions of the file by providing the byte start and end locations of each result. We run ANNIE on the files and then use the annotation differential tool to export them to HTML files based on the category specified in the question document. Sentences that do not contain the type of information required by the question are discarded.

GATE is unable to identify many types of named entities that would match answers to questions asked by the QA Track 2008. For example, the track question might be asking for a "reason" but GATE is not capable of determining portions of a sentence that match "reason", which hurt our QA system performance.

4. Parsing tree matching and scoring

Now the candidate sentences have the same focus as the questions, and also contain the type of information asked by the questions. In this step we will match the parsing tree of the candidate sentence and the question and generate a score. The candidate sentences are ranked according to their scores. Answer strings are extracted from all the sentences whose scores are above a preset threshold.

The tree matching and scoring process is shown in Figure 3. The process starts with an edge matching. If any edges match then the edges are recursively linked together if possible. Finally scores are calculated for the largest possible structure and any remaining nodes should they exist.

For each candidate sentence

Load its parsing tree;

Search for the common nodes between the parsing trees of candidate sentence and question;

Search for the common edges between the parsing trees of candidate sentence and question;

Recursively search for the common large structures (including more than one edge) between the parsing trees of candidate sentence and question;

Assign score to all common nodes, edges, and structures, calculate the total score;

Rank each sentence according to its total score;

Figure 3 Sentence matching and score process

3. Conclusion and Future Work

In this paper we discuss our Question Answering system developed for the TAC QA track 2008. Question answering is not a single research task, instead its performance is affected by many processes involved. The key component is our system is the Tree Matching step to rank the candidate sentences according to its structural similarity to questions. This approach may be more effective in large corpus where the same information is expressed in different ways hence improve the probability of structural matching.

The future work includes the development of our own Named Entity Recognition software since categories used in GATE are too coarse.

4. Acknowledgement

This work is partially funded by Scholar Academy at the University of Houston-Downtown.

Reference

1. BLOG06 corpus, http://ir.dcs.gla.ac.uk/test_collections
2. BOOST C++ Library, http://www.boost.org/
3. GATE Natural Language Processing package, http://gate.ac.uk/ie/
4. Xin Li, Dan Roth, Learning Question Classifiers. COLING'02, Aug., 2002.
5. D. Lin, Dependency-based Evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems, Granada, Spain, May, 1998.
6. SLTagger, http://www-tsujii.is.s.u-tokyo.ac.jp/uima/
7. Text Analysis Conference 2008 Question Answering Track, http://www.nist.gov/tac/tracks/2008/qa/