Graph-based Approach to the Question Answering Task Based on Entrance Exams

Helena Gómez-Adorno[1], Grigori Sidorov[1], David Pinto[2], and Alexander Gelbukh[1]

[1]Centro de Investigación,
Instituto Politécnico Nacional,
México D.F., Mexico
helena.adorno@gmail.com,
sidorov@cic.ipn.mx, www.gelbukh.com

[2]Facultad de Ciencias de la Computación,
Benemérita Universidad Autónoma de Puebla,
Puebla, Mexico
dpinto@cs.buap.mx

Abstract. This paper describes the approach used in the system for the Question Answering Task based on Entrance Exams, which was presented at the CLEF 2014. The task aims to evaluate methods of text understanding with reading comprehension tests. The system should read a given document and answer multiple-choice questions about it. Our approach transforms the documents along with the multiple-choice answers into a graph-based representation that contains lexical, morphological, and syntactic features. After this, it traverses different paths both in the document itself and in the the graphs of the answers in order to find these features of the graphs. It is performed by counting text components: lemmas, PoS tags, grammatical tags. As the result of this procedure, the system constructs several feature vectors: one for each traversed graph. Finally, a cosine based similarity is calculated over these feature vectors in order to rank the multiple-choice answers and select the correct one—with the best similarity with the graph that corresponds to the text itself. Our system obtained a c@1 of 0.375, which was outperformed only by one system in the competition.

Keywords: Question answering system, reading comprehension, entrance exams, graph-based representation, graph similarity, extraction of features from graphs

1 Introduction

In this paper we present the experiments carried out as part of the participation in the Question Answering track based on Entrance Exams presented at the CLEF 2014. The Entrance Exam task was proposed in 2013 as a pilot task [1] in

the Question Answering for Machine Reading Evaluation Lab (QA4MRE), which was offered at the Conference and Labs of the Evaluation Forum (CLEF) since 2011 [2, 3]. The entrance exams task evaluates systems in the same situation, in which high school students are evaluated for entering a university. The challenge consists of reading a document and identifying a correct answer (from multiple choices) for a set of questions about the information that is expressed or implied in the text. The questions are written in the form of multiple choices; each question has 4 different options, and only one option is the correct answer. Exams are created by the Japanese National Center for University Admissions Tests. The Entrance Exams corpus is provided by NII's Todai Robot Project[1] and NTCIR[2].

Since the first edition of QA4MRE task in 2011, and later in the 2012 and 2013, a single evaluation platform for the experimentation with new techniques and methodologies for this problem has provided. In this sense, we can take the systems presented at this conference as state of the art work in this research field.

The rest of the paper is organized as follows. Section 2 describes our approach and the system architecture. Section 3 presents the configuration of the submitted runs and the evaluation results. Finally, Section 4 presents the conclusions and outlines some directions of future work.

2 System Architecture

For many problems in natural language processing, graph structure is an intuitive, natural and direct way to represent data. There exist several research works that have employed graphs for text representation in order to solve some particular problem [4]. We propose an approach based on a graph methodology for document understanding, which is described in detail before in [5], and built the corresponding system. The system consists of the following submodules: document preprocessing, graph generation and answer validation. The general architecture is illustrated in Figure 1.

2.1 Document Preprocessing

First we perform document (pre)processing. An XML parser receives as input a structured corpus in XML format as it is shown in Figure 1. This XML file contains all the documents, along with their respective questions and multiple choice answers. An XML interpreter extracts the documents, questions and associated answers. It stores the questions and answers identifying them according to the document, to which they belong, in order to be used in further processing. Further, the questions associated to each document are analyzed, identifying the "question keywords" (*what*, *where*, *when*, *who*, etc.), and the result is passed to

[1] http://21robot.org/About/
[2] http://research.nii.ac.jp/ntcir/index-en.html

the next module. After this, *hypothesis generation* module formulates several candidate "answer hypotheses" as the modified versions of the original question, replacing these words with one of the possible answers given in the test data. For example, given the question: ***Who*** *is the founder of the SING campaign?* and the possible answer: ***Annie Lennox***. The obtained hypothesis is: ***Annie Lennox*** *is the founder of the SING campaign.*

Afterwards, we perform anaphora resolution in the documents using the JavaRAP[3] system. It was observed that applying anaphora resolution in QA systems improves the precision [6].

The output of this module is the set of answer hypotheses along with their reference documents.

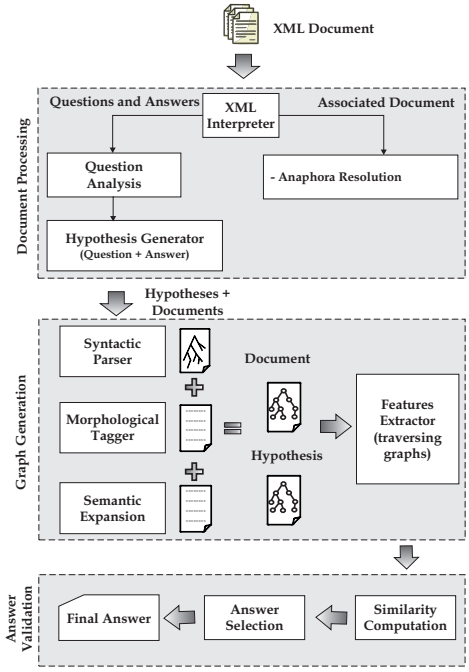

Fig. 1. Graph-based system architecture

2.2 Graph Generation Module

In the graph generation module, text documents along with their hypotheses are parsed to produce their graph-based representation. For the graph representation we took into account various linguistic levels (lexical, syntactic, morphological

[3] http://wing.comp.nus.edu.sg/ qiu/NLPTools/JavaRAP.html

and semantic relationships) in order to capture the majority of features present in the text.

The process of the graph generation is performed by the following submodules:

Syntactic Parser is the base of the graph structure. We use the Stanford Dependency Parser[4] for producing the parsed tree for each sentence of the documents. In this type of parsing, we detect grammatical relation between words of the sentences.

Morphological Tagger obtains PoS tags of the words. For this, we used the Stanford Log linear Part-Of-Speech Tagger[5] for English. The Lancaster stemmer algorithm was used in order to obtain word stems.

Semantic Expansion uses the Wordnet taxonomy [7] in order to add semantic relations between nodes of the graphs, i.e., the graph is expanded using the taxonomy: some nodes are added.

As the result of this process, each document is represented as a tree rooted in a $ROOT - 0$ node. The vertices to sub-trees represent all sentences in the document. The nodes of the trees represent word or lemmas of the sentences along with their part-of-speech tags. The vertices between nodes represent the dependency tags between these connected nodes and the frequency label shows the number of occurrences of the pair (initial_node, final_node) in the graph plus the frequency of the dependency tag of the same pair of nodes. In the same way all answer hypotheses are represented as trees with the same characteristics.

In Figure 2 we present the graph-based representation for the hypothesis "Annie Lennox is the founder of the SING campaign", whereas Figure 3 shows the graph-based representation for the first three sentences of the reference document associated to this question.

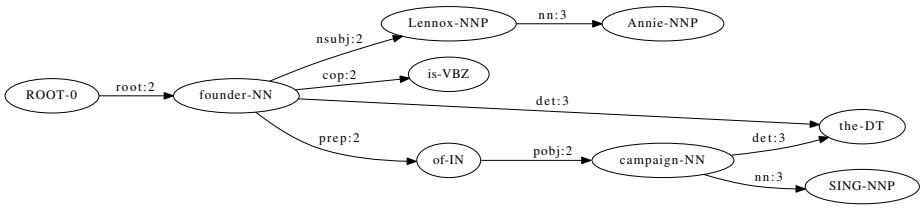

Fig. 2. Graph-based representation of the hypothesis "Annie Lennox is the founder of the SING campaign"

The feature extraction starts by fixing the root node of the hypothesis graph as the initial node, whereas the selected final nodes correspond to the rest nodes

[4] http://nlp.stanford.edu/software/lex-parser.shtml
[5] http://nlp.stanford.edu/software/tagger.shtml

of the hypothesis graph. We use the *Dijkstra's Algorithm* [8] for finding the shortest paths between the initial and each final node. After this, we count the occurrences of all the multi-level linguistic features considered in the text representation such as PoS tags and dependencies tags found in the path. The same procedure is performed with the document graph, using the pair of nodes identified in the hypothesis as the initial and final node. As the result of this procedure, we obtain two feature vectors: one for the answer hypothesis and another one for the reference document. This module was implemented in Python, using the NetworkX[6] package for creation and manipulation of graphs.

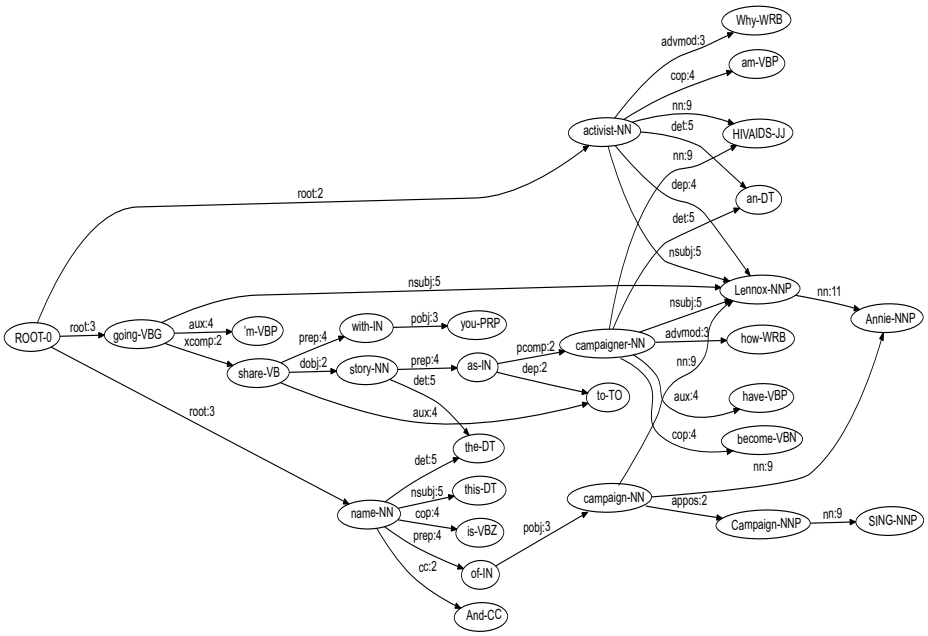

Fig. 3. Example of the graph-based representation of a reference document (3 first sentences

2.3 Answer Validation Module

This module receives several feature vectors ($\overrightarrow{f_{t,i}}$) for each text. Thus, the reference document d is now represented by m features ($d^* = \{\overrightarrow{f_{d,1}}, \overrightarrow{f_{d,2}}, ..., \overrightarrow{f_{d,m}}\}$), as well as the answer hypothesis h, ($h^* = \{\overrightarrow{f_{h,1}}, \overrightarrow{f_{h,2}}, ..., \overrightarrow{f_{h,m}}\}$), being m the number of different paths that can be traversed in both graphs.

[6] https://networkx.github.io/

We use the following cosine similarity measure for calculating the degree of similarity in each traversed path:

$$\begin{aligned} Similarity(h^*, d^*) &= \sum_{i=1}^{m} Cosine(\overrightarrow{f_{h,i}}, \overrightarrow{f_{d,i}}) \\ &= \sum_{i=1}^{m} \frac{\overrightarrow{f_{h,i}} \cdot \overrightarrow{f_{d,i}}}{||\overrightarrow{f_{h,i}}|| \cdot ||\overrightarrow{f_{d,i}}||} \\ &= \sum_{i=1}^{m} \frac{\sum_{j=1}^{|V|} (f_{(h,i),j} \times f_{(d,i),j})}{\sqrt{\sum_{j=1}^{|V|} (f_{(h,i),j})^2} \times \sqrt{\sum_{j=1}^{|V|} (f_{(d,i),j})^2}}. \end{aligned} \quad (1)$$

After obtaining all similarity scores of the four hypotheses for each question, the hypothesis achieving the highest score is selected as the correct answer. We answered every question of the task using this methodology.

3 Experimental Results

This section describes the data sets of the challenge used for evaluation of the systems. Additionally, the results obtained in the experiments are reported and discussed.

Following 2013 edition, the test set 2014 based on Entrance Exams was composed of tests for reading comprehension taken from the Japanese Center Test, which is a nation-wide achievement test for admission in Japanese universities.

The organizers released two different datasets, one for training and one for testing. Both data sets were composed of the following elements:

- 12 test documents,
- 60 questions (5 questions for each document),
- 240 choices/options (4 for each question).

The main measure used in this evaluation is c@1, which is defined as shown in equation 2. This measure was defined in the QA4MRE task at CLEF 2011 with the purpose of allowing the systems to decide whether or not to answer a given question. The aim of this procedure is to reduce the amount of incorrect answers, maintaining the number of correct ones, i.e., a system is penalized for answering incorrectly and it is better not to answer at all if not sure:

$$c@1 = \frac{1}{n}(n_R + n_U \frac{n_R}{n}), \quad (2)$$

where:

n_R: number of correctly answered questions,
n_U: number of unanswered questions,
n: total number of questions.

Table 1 presents the comparative results obtained in the QA 2014 competition for Entrance Exams. Approximately thirty runs were submitted to the

competition from various research teams. We submitted eight runs with different configurations of the system. The run that ranked highest was *cicnlp-8*, which obtained a c@1 of 0.375 answering 21 questions correctly. At the reading comprehension level the run *cicnlp-8* only passed 3 out of 12 tests, whereas the run *cicnlp-2* passed 4 out of 12 tests. Even though the run *cicnlp-2* passed one more test than our best run, it only obtained a c@1 of 0.303. In the last column of Table 1 we show the quantity of tests passed by the runs (tests with c@1 $>$ 0.5).

Table 1. Results of participation in QA task 2014 based on Entrance Exams

Run	Correctly Answered	Incorrectly Answered	Unanswered	c@1	Tests with c@1 > 0.5
cicnlp-1	16	40	0	0.285	3/12
cicnlp-2	19	37	0	0.339	2/12
cicnlp-3	17	39	0	0.303	4/12
cicnlp-4	17	39	0	0.303	2/12
cicnlp-5	13	43	0	0.232	1/12
cicnlp-6	16	40	0	0.285	3/12
cicnlp-7	20	36	0	0.357	2/12
cicnlp-8	**21**	**35**	**0**	**0.375**	3/12

Table 2 shows the features included in the graph-based representation and which of those features were used in the feature extraction technique for each of the eight configurations. Those configurations were selected during the evaluation process with the training dataset released by the organizers.

The *cicnlp-1* run includes the words and the POS tags in the nodes of the graph-based representation, as well as the dependency tags, which represents the relation between each pair of nodes (vertices). Note that in this run we do not use lemmas in graph nodes and do not consider frequencies for the vertices. The features extracted in this run are PoS tags and dependency tags. In the case of the *cicnlp-2* the graph representation is the same, but we add the word count to the extracted feature set. In case of the runs *cicnlp-1* and *cicnlp-2*, we used an algorithm for obtaining all shortest path between an initial and a final node. The rest of the runs use the Dijkstra algorithm for obtaining only one shortest path.

The *cicnlp-3* run includes the frequency count of the vertices (initial node to final node + dependency tags) in the same way as it is explained in Section 2.2. The *cicnlp-4* uses the same graph representation and extracted features than run *cicnlp-2*, the only difference is the traversal algorithm.

The *cicnlp-5* and *cicnlp-6* runs apply the feature extraction technique when the words in the hypothesis graph are expanded with their corresponding set of synonyms (without applying the process of word sense disambiguation). The run *cicnlp-6* uses stems instead of the full words in the graph-based representations. The features extracted in both runs are words (count), PoS tags (count)

Table 2. Configurations of the graphs representation and the extracted features used in the runs.

	Representation schemes							Extracted features		
Run	**Words**	**POS tags**	**Dependency tags**	**Stemming**	**Frequency**	**Synonym expansion**	**Hypernym expansion**	**Words count**	**POS tags count**	**Dependency tags count**
cicnlp-1	✓	✓	✓						✓	✓
cicnlp-2	✓	✓	✓					✓	✓	✓
cicnlp-3	✓	✓	✓		✓				✓	✓
cicnlp-4	✓	✓	✓					✓	✓	✓
cicnlp-5	✓	✓	✓			✓		✓	✓	✓
cicnlp-6	✓	✓	✓	✓		✓		✓	✓	✓
cicnlp-7	✓	✓	✓				✓		✓	✓
cicnlp-8	✓	✓	✓	✓			✓		✓	✓

and dependency tags (count). The runs *cicnlp-7* and *cicnlp-8* apply the feature extraction technique when the words in the hypothesis graph are expanded with their corresponding set of hypernyms (without applying the process of word sense disambiguation). The run *cicnlp-8* uses stems instead of the full words in the graph-based representations. The features extracted in both runs are PoS tags (count) and dependency tags (count).

We found that as far as the reading level is concerned, different runs were able to pass different tests. For example, the run *cicnlp-8* passed the tests 13, 16 and 18; the run *cicnlp-3* passed the tests 13, 14, 15 and 16; the run *cicnlp-6* passed the tests 13, 16 and 23. In future, we plan to combine different runs, and in this manner we would be able to pass 6 out of 12 tests. Besides, the run *cicnlp-8* passed the test 13 correctly answering all questions.

4 Conclusion and Future Work

We described the approach and the system developed as a part of our participation of QA task 2014 based on Entrance Exams. The approach uses a graph structure for representing the documents and the answer hypotheses. It extracts linguistic features from both graphs—documents and answer hypotheses—by traversing shortest paths. The features are further used for computing the similarity between the document and the answer hypotheses.

We sent eight runs to the competition. The best run (*cicnlp-8*) of our system achieves a c@1 of 0.375, which was outperformed only by one system.

For future work, we hope that the use of domain-specific techniques of question answering will improve the performance of the algorithm for this particular problem. Textual entailment, named entity recognition, analysis of the type of question can improve the final results of our system in the task.

References

1. Peñas, A., Miyao, Y., Forner, P., Kando, N.: Overview of QA4MRE 2013 Entrance Exams task. In: CLEF (Online Working Notes/Labs/Workshop). (2013)
2. Peñas, A., Hovy, E.H., Forner, P., Rodrigo, Á., Sutcliffe, R.F.E., Forascu, C., Sporleder, C.: Overview of QA4MRE at CLEF 2011: Question Answering for Machine Reading Evaluation. In: CLEF (Notebook Papers/Labs/Workshop). (2011)
3. Peñas, A., Hovy, E.H., Forner, P., Rodrigo, Á., Sutcliffe, R.F.E., Sporleder, C., Forascu, C., Benajiba, Y., Osenova, P.: Overview of QA4MRE at CLEF 2012: Question Answering for Machine Reading Evaluation. In: CLEF (Online Working Notes/Labs/Workshop). (2012)
4. Mihalcea, R., Radev, D.: Graph-based natural language processing and information retrieval. Cambridge University Press (2011)
5. Pinto, D., Gómez-Adorno, H., Ayala, D.V., Singh, V.K.: A graph-based multi-level linguistic representation for document understanding. Pattern Recognition Letters **41** (2014) 93–102
6. Vicedo, J.L., Ferrandez, A.: Importance of pronominal anaphora resolution in question answering systems. In: Proceedings of the 38th Annual Meeting of the Association for Computational Linguistics (ACL 2000). (2000) 555–562
7. Miller, G.A.: WordNet: A lexical database for English. Communications of the ACM **38** (1995) 39–41
8. Dijkstra, E.W.: A note on two problems in connexion with graphs. Numerische mathematik **1**(1) (1959) 269–271