CICBUAPnlp: Graph-Based Approach for Answer Selection in Community Question Answering Task

Helena Gómez-Adorno, Grigori Sidorov
Center for Computing Research
Instituto Politécnico Nacional
Av. Juan de Dios Bátiz
C.P. 07738, Mexico City, Mexico
helena.adorno@gmail.com
sidorov@cic.ipn.mx

Darnes Vilariño, David Pinto
Faculty of Computer Science
Benemérita Universidad Autónoma de Puebla
Av. San Claudio y 14 sur
C.P. 72570, Puebla, Mexico
darnes@cs.buap.mx
dpinto@cs.buap.mx

Abstract

This paper describes our approach for the Community Question Answering Task, which was presented at the SemEval 2015. The system should read a given question and identify good, potentially relevant, and bad answers for that question. Our approach transforms the answers of the training set into a graph based representation for each answer class, which contains lexical, morphological, and syntactic features. The answers in the test set are also transformed into the graph based representation individually. After this, different paths are traversed in the training and test sets in order to find relevant features of the graphs. As a result of this procedure, the system constructs several vectors of features: one for each traversed graph. Finally, a cosine similarity is calculated between the vectors in order to find the class that best matches a given answer. Our system was developed for the English language only, and it obtained an accuracy of 53.74 for subtask A and 44.0 for subtask B.

1 Introduction

In this paper we present the experiments carried out as part of our participation in the SemEval-2015 Task 3 (Answer Selection in Community Question Answering). The Answer Selection in Community Question Answering task is proposed for the first time this year in the International Workshop on Semantic Evaluation (SemEval-2015). The task is based on an application scenario, which is related to textual entailment, semantic similarity and NL inference.

Community question answering (CQA) websites enable people to post questions and answers in various domains. In this way, users can obtain specific answers to their questions, instead of searching in the large volume of information available in the web. However, it takes effort to go through all possible answers and select which one is the most accurate one for a specific question. The task proposes to automate this process by predicting the quality of existing answers with respect to a question.

There are few works in the literature on evaluating the quality of answers provided in CQA sites. Most of such works employ non-textual and temporal features in order to built classification models for predicting the best answer for a given question. In (Jeon et al., 2006), the authors extract 13 non-textual features from the Naver data set and build a maximum entropy classification model to predict the quality (three classes: Bad, Medium and Good) of a given answer. A similar approach is used in (Shah and Pomerantz, 2010), but extracting 21 features (mainly non-textual) from *Yahoo! Answers*; the authors employ a logistic regression and classification model to predict the best answer. Besides, a set of temporal features is proposed in (Cai and Chakravarthy, 2011) in order to predict the best answer for a given question. In this work the authors argue that the traditional classification approaches are not well suited for this problem because of the highly imbalanced ratio of the best answer and the non-best answers in their data set, so they propose to use learning to rank approaches.

Unlike these approaches, we use only textual information for predicting the quality of the answers.

Our approach is based on our previous research (Pinto et al., 2014) and (Sidorov et al., 2014), where we propose the graph-based representation model (Integrated Syntactic Graph) and the soft similarity measure (soft cosine measure). Our experimental results are promising, they overcome the baseline system for this challenge.

The rest of the paper is organized as follows. Section 2 describes our approach. 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 Approach

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 (Mihalcea and Radev, 2011). We propose an approach based on a graph methodology, which was described in detail in (Pinto et al., 2014), for building the corresponding system of the two subtasks. These subtasks are described as follows:

Subtask A Given a question (short title + extended description) and a list of community answers, classify each of the answers as: Good, Potential or Bad (bad, dialog, non-English, other).

Subtask B Given a YES/NO question (short title + extended description) and a list of community answers, decide whether the global answer to the question should be yes, no or unsure, based on the individual good answers.

The proposed system consists of the following submodules: document preprocessing, graph generation, and answer quality classification.

2.1 Document Preprocessing

An XML parser receives as input a structured corpus in XML format. This XML file contains all the questions, along with their respective answers. An XML interpreter extracts the questions and associated answers.

Thereafter, we process the answers for both subtasks separately. All the answers belonging to the same class are grouped together, and the result is passed to the next module. This means that at the end of this module, we will have all the good answers in one document, the bad ones in another document and so on for all classes. In the same way, for the task B, the yes/no answers are grouped together in different documents.

2.2 Graph Generation

In the graph generation module, all sentences of each class are parsed to produce what we call their Integrated Syntactic Graph (ISG) representation (see (Pinto et al., 2014)). For the graph representation we took into account various linguistic levels (lexical, syntactic, morphological, and semantic) in order to capture the majority of the features present in the text.

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

The Syntactic Parser is the base of the graph structure. We use the Stanford Dependency Parser[1] for producing the parsed tree for each sentence of the documents. In this type of parsing, we detect grammatical relation.

The Morphological Tagger obtains PoS tags of words. For this purpose we used the Stanford Log linear Part-Of-Speech Tagger[2] for English. The Lancaster stemmer algorithm was used in order to obtain word stems.

As a result of this process, each class is represented as a graph rooted in a $ROOT - 0$ node. The vertices to sub-trees represent all sentences in the class document. The nodes of the trees represent words or lemmas of the sentences along with their part-of-speech tags. The vertices between nodes represent the dependency tags between these connected nodes along with a frequency label, for example: *nsubj-5*, that 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, the answers to be classified in one of the quality classes are represented in an ISG with the same characteristics.

In order to fully understand the process of construction of the ISG and the collapse of nodes in the

[1]http://nlp.stanford.edu/software/lex-parser.shtml

[2]http://nlp.stanford.edu/software/tagger.shtml

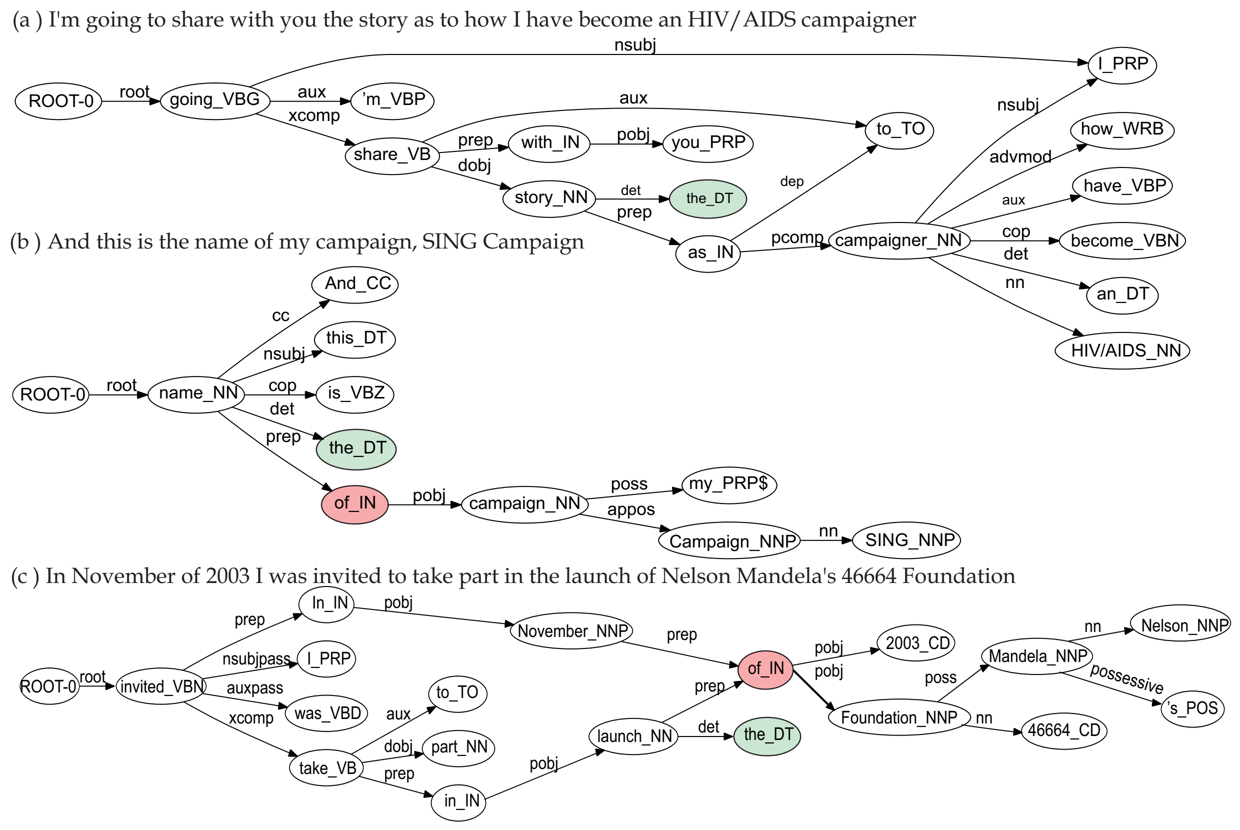

Figure 1: Dependency trees of three sentences of a target text using word POS combination for the nodes and dependency labels for the edges

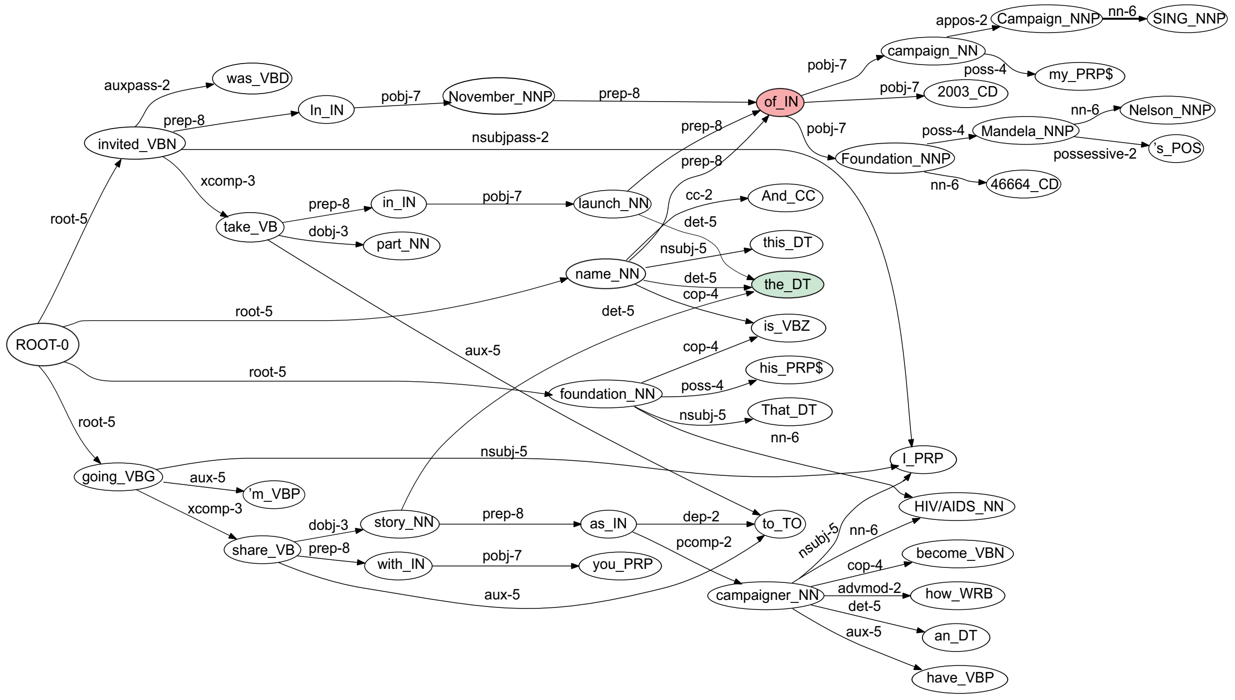

Figure 2: The Integrated Syntactic Graph for the three sentences considered as example

graph, in Figure 1, we show the dependency trees of three sentences; each node of the graph is augmented with other annotations, such as the combination of lemma (or word) and POS tags: (lemma POS).

The collapsed graph of the three sentences is shown in Figure 2. Each edge of this graph contains the dependency tag together with a number that indicates the frequency of the dependency tag plus the frequency of the pair of nodes, both calculated using the occurrences of the dependency trees associated to each sentence.

The feature extraction process starts by fixing the root node of the answer graph as the initial node, whereas the selected final nodes correspond to the remaining nodes of the answer graph. We use the $Dijkstra's\ Algorithm$ (Dijkstra, 1959) 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 dependency tags found in the path. The same procedure is performed with the class document graph, using the pair of nodes identified in the answer graph as the initial and final node. As a result of this procedure, we obtain two feature vectors: one for the answer and another one for the class document. This module was implemented in Python, using the NetworkX[3] package for creation and manipulation of graphs.

2.3 Classification based on Quality of Answers

This module receives several feature vectors ($\overrightarrow{f_{t,i}}$) for each class document. Thus, the class 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 different answers a, ($a^* = \{\overrightarrow{f_{a,1}}, \overrightarrow{f_{a,2}}, ..., \overrightarrow{f_{a,m}}\}$), being m the number of different paths that can be traversed in both graphs.

We use the cosine similarity measure from the equation below for calculating the degree of similarity among each traversed path.

$$Similarity(a^*, d^*) = \sum_{i=1}^{m} Cosine(\overrightarrow{f_{a,i}}, \overrightarrow{f_{d,i}})$$

$$= \sum_{i=1}^{m} \frac{\overrightarrow{f_{a,i}} \cdot \overrightarrow{f_{d,i}}}{||\overrightarrow{f_{a,i}}|| \cdot ||\overrightarrow{f_{d,i}}||}$$

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

After obtaining all similarity scores between the answers with each of the class documents, the class (to which the document belongs) achieving the highest score is selected as the correct class for each answer.

3 Results

The acronym of our system is CICBUAPnlp. Tables 1 and 2 show the scores for the English subtasks A and B on the test data, respectively. Although, our results did not overcome the general average, it is worth noting that our methodology is quite simple and straightforward. We only used syntactic and morphological features, thus comparing the structures of the answers against the structure of the labeled sets. Instead of training a classifier, we built a Syntactic Integrated Graph for each class and then try to match the answers in the test set against them, calculating in this way the similarity between the graphs.

Table 1: Results of the subtask A, English

TeamId	Macro F1	Accuracy	Rank
JAIS	57.19	72.52	1
HITSZ-ICRC	56.41	68.67	2
QCRI	53.74	70.50	3
ECNU	53.47	70.55	4
ICRC-HIT	49.60	67.68	5
VectorSlu	49.10	66.45	5
Shiraz	47.34	56.83	7
FBK-HLT	47.32	69.13	8
Voltron	46.07	62.35	9
CICBUAPnlp	**40.40**	**53.74**	**10**
Yamraj	37.65	45.50	11
CoMiC	30.63	54.20	12

4 Conclusion and Future Work

We described the approach and the system developed as a part of our participation in the Answer Selection in Community Question Answering task. The approach uses a graph structure for representing the classes and the answers. It extracts linguistic features from both graphs—classes and answers—by traversing shortest paths. The features

Table 2: Results of the subtask B, English

TeamId	Macro F1	Accuracy	Rank
VectorSlu	63.7	72.0	1
ECNU	55.8	68.0	2
QCRI	53.6	64.0	3=4
HITSZ-ICRC	53.6	64.0	3=4
CICBUAPnlp	**38.8**	**44.0**	**5**
ICRC-HIT	30.9	52.0	6
Yamraj	29.8	28.0	7
FBK-HLT	27.8	40.0	8

are further used for computing the similarity between the classes and the answers.

We sent two runs (primary and contrastive) for each English subtask to the evaluation forum. The best run in both cases was the primary run.

In future work, we are planning to use the soft cosine measure to compare the similarity between the answers and the quality classes, thus evaluating the feasibility of this kind of structures for this task.

Acknowledgments

This work was done under partial support of the Mexican Government (CONACYT, SNI, COFAA-IPN, SIP-IPN 20144534, 20144274) and FP7 -PEOPLE-2010-IRSES: "Web Information Quality Evaluation Initiative (WIQ-EI)" European Commission project 269180.

References

Yuanzhe Cai and Sharma Chakravarthy. 2011. Predicting answer quality in q/a social networks: Using temporal features. Technical report, University of Texas at Arlington.

Edsger. W. Dijkstra. 1959. A note on two problems in connexion with graphs. *Numerische Mathematik*, 1:269–271.

Jiwoon Jeon, W. Bruce Croft, Joon Ho Lee, and Soyeon Park. 2006. A framework to predict the quality of answers with non-textual features. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 228–235, New York, NY, USA. ACM.

Rada Mihalcea and Dragomir Radev. 2011. *Graph-based natural language processing and information retrieval*. Cambridge; New York.

David Pinto, Helena Gómez-Adorno, Darnes Vilariño, and Vivek Kumar Singh. 2014. A graph-based multi-level linguistic representation for document understanding. *Pattern Recognition Letters*, 41(0):93 – 102. Supervised and Unsupervised Classification Techniques and their Applications.

Chirag Shah and Jefferey Pomerantz. 2010. Evaluating and predicting answer quality in community qa. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 411–418, New York, NY, USA. ACM.

Grigori Sidorov, Alexander F. Gelbukh, Helena Gmez-Adorno, and David Pinto. 2014. Soft similarity and soft cosine measure: Similarity of features in vector space model. *Computación y Sistemas*, 18(3):491 – 504.