

Using Recognizing Textual Entailment as a Core Engine for Answer Validation

Rui Wang¹ and Günter Neumann²

¹ Saarland University, Saarbrücken, Germany, rwang@coli.uni-sb.de

² LT lab, DFKI, Saarbrücken, Germany, neumann@dfki.de *

Abstract. This paper is about our approach to answer validation, which centered by a *Recognizing Textual Entailment* (RTE) core engine. We first combined the question and the answer into *Hypothesis* (**H**) and view the document as *Text* (**T**); then, we used our RTE system to check whether the entailment relation holds between them. Our system was evaluated on the *Answer Validation Exercise* (AVE) task and achieved f-measures of 0.46 and 0.55 for two submission runs, which both outperformed others' results for the English language.

1 Introduction and Related Work

Question Answering (QA) is an important task in *Natural Language Processing* (NLP), which aims to mine answers to natural language questions from large corpora. *Answer validation* (AV) is to evaluate the answers obtained by the former stages of a QA system and select the most proper answers for the final output.

In recent years, a new trend is to use RTE (Dagan et al., 2006) to do answer validation, cf. the AVE 2006 Working Notes (Peñas et al., 2006). Most of the groups use lexical or syntactic overlapping as features for machine learning; other groups derive the logic or semantic representations of natural language texts and perform proving.

We also developed our own RTE system, which proposed a new sentence representation extracted from the dependency structure, and utilized the Subsequence Kernel method (Bunescu and Mooney, 2006) to perform machine learning. We have achieved good results on both the RTE-2 data set (Wang and Neumann, 2007a) and the RTE-3 data set (Wang and Neumann, 2007b), especially on *Information Extraction* (IE) and QA pairs.

Therefore, the work we have done has two motivations: 1) to improve answer validation by using RTE techniques; and 2) to further test our RTE system in concrete NLP applications. The following of the paper will start with introducing our AVE system, which consists of the preprocessing part, the RTE core engine, and the post-

* The work presented here was partially supported by a research grant from the German Federal Ministry of Education, Science, Research and Technology (BMBF) to the DFKI project HyLaP (FKZ: 01 IW F02) and by the EU-funded project QALL-ME (FP6 IST-033860).

processing part. Then, the results of our two submission runs will be shown, followed by a discussion on error sources. In the end, we will summarize our work.

2 Our RTE-based AVE System

Our AVE system uses an RTE system (*TERA* – Textual Entailment Recognition for Application) as a core engine. The preprocessing module mainly adapts questions, their corresponding answers, and supporting documents into *Text(T)-Hypothesis(H)* pairs, assisted by manually designed patterns. The post-processing module will validate each answer and select a most proper one based on the output of the RTE system.

2.1 Preprocessing

The given input of the AVE task is a list of questions, their corresponding answers and the documents containing these answers. Usually, we need to validate several answers for each question. For instance, for the question, “*In which country was Edouard Balladur born?*” the QA system gives out several candidate answers to this question, “*Frances*”, “*12% jobless rate*”, or “*7*”, and also supporting documents where the answers come from. Here, the assumption for the validation process is that *if the answer is to the question, the document which contains the answer should entail the statement derived by combining the question and the answer.*

In order to combine a question and an answer into a statement, we manually constructed some language patterns for the input questions. As for the question given above, we will apply the following pattern, “*Edouard Balladur was born in <Answer>*”, and substitute the “*<Answer>*” by each candidate answer to form an **H** – a hypothesis. Since the supporting documents are naturally the **Ts** – texts, the **T-H** pairs are built up accordingly, and these **T-H** pairs can be the input for any generic RTE systems.

2.2 The RTE Core Engine

The RTE core engine contains a main approach with two backup strategies (Wang and Neumann, 2007a). In brief, the main approach firstly extracts common nouns between **T** and **H**; then it locates them in the dependency parse tree as *Foot Nodes* (FNs). Starting from the FNs, a common parent node can be found in each tree, which will be named as *Root Node* (RN); Altogether, FNs, the RN, and the dependency paths in-between will form a *Tree Skeleton* (TS) for each tree. On top of this feature space, we can apply subsequence kernels to represent these TSs and perform kernel-based machine learning to predict the final answers discriminatively.

The backup strategies will deal with the **T-H** pairs which cannot be solved by the main approach. One backup strategy is called Triple Matcher, as it calculates the

overlapping ratio on top of the dependency structures in a triple representation*; the other is simply a Bag-of-Words (BoW) method, which calculates the overlapping ratio of words in **T** and **H**.

2.3 Post-processing

The RTE core engine has given us: 1) for some of the **T-H** pairs, we directly know whether the entailment holds; 2) every **T-H** pair has a triple similarity score and a BoW similarity score. If the **T-H** pairs are covered by our main approach, we will directly use the answers; if not, we will use a threshold to decide the answer based on the two similarity scores. In practice, the thresholds are learned from the training corpus.

For adapting the results back to the AVE task, the “YES” entailment cases will be the validated answers and the “NO” entailment cases will be the rejected one. In addition, the selected answers (i.e. the best answers) will naturally be the pairs covered by our main approach or (if not,) with the highest similarity scores.

3 Results and Error Analysis

We have submitted two runs for AVE2007. Both of the two runs we have used the main approach plus one backup strategy. In the first run, the BoW similarity score was the backup, while in the second run, the triple similarity score was taken. We have used *Minipar* (Lin, 1998) as our dependency parser and our machine learning process was performed by the classifier SMO from the WEKA toolkit (Witten and Frank, 1999). The following Table 1 shows the results,

Table 1. Results of our two submission runs.

Submission Runs	Recall	Precision	F-measure	QA Accuracy
dfki07-run1.txt	0.62	0.37	0.46	0.16
dfki07-run2.txt	0.71	0.44	0.55	0.21

Though the absolute scores are not very promising, they are still better than all the others’ results for the English language this year. The second run outperforms the first run in all respects, which shows advantages of the triple similarity score. The gold standard does not contain the “*SELECTED*” answers, thus, we will not discuss the QA accuracy here. Instead, the error analysis will focus on the loss of recall and precision.

As for recall, among all the errors, half of them belong to one type. For questions like “*What is the occupation of Kiri Te Kanawa?*” we have used the pattern “*The occupation of Kiri Te Kanawa is <Answer>*”, which has caused problems, because “*occupation*” usually did not appear in the documents. Instead, a pattern like “*Kiri Te*

* A triple is of the form <node1, relation, node2>, where node1 represents the head, node2 the modifier, and relation the dependency relation.

Kanawa is <Answer>” might be much better. Some other errors are from the noise of web documents, on which the dependency parser could not work very well.

The precision of our two runs are rather poor. After taking a closer look at the errors, we have found that most of the errors also belong to one type. In those answer-document pairs (e.g. id=119_2, id=125_1, id=133_1, etc.), the answers are usually very long, which consist of a large part of the documents. Some extreme cases (e.g. id=112_2, id=172_2, etc.), the answers are very long and exactly the same as the documents. Due to the characteristics of our method (i.e. using RTE for AVE), these answers will get high similarity scores, which are wrongly validated.

In addition, some other errors like trivial answers (e.g. “one”) could be avoided by adding some rules. As a whole, more fine-grained classification of answers could be helpful to improve the system.

4 Conclusion and Future Work

In conclusion, we have described our approach to answer validation using RTE as a core engine. On the one hand, it is an effective way to do the answer validation task; on the other hand, it is also a promising application for our developed RTE system. The results have shown the advantages of our combination of the main approach and backup strategies.

After error analysis, the possible future directions are: 1) preprocessing the documents to clean the noisy web data; 2) making the patterns be automatically generated; 3) utilizing question analysis tools to acquire more useful information.

References

1. Bunescu, R. and Mooney, R. 2006. Subsequence Kernels for Relation Extraction. In *Advances in Neural Information Processing Systems 18*. MIT Press.
2. Dagan, I., Glickman, O., and Magnini, B. 2006. The PASCAL Recognising Textual Entailment Challenge. In Quiñero-Candela et al., editors, *MLCW 2005, LNAI Volume 3944*, pages 177-190. Springer-Verlag.
3. Lin, D. 1998. Dependency-based Evaluation of MINIPAR. In *Workshop on the Evaluation of Parsing Systems*.
4. Peñas, A., Rodrigo, Á., Sama, V., and Verdejo, F. 2006. Overview of the Answer Validation Exercise 2006. In *AVE 2006 Working Notes*.
5. Wang, R. and Neumann, G. 2007a. Recognizing Textual Entailment Using a Subsequence Kernel Method. In *Proc. of AAAI 2007*.
6. Wang, R. and Neumann, G. 2007b. Recognizing Textual Entailment Using Sentence Similarity based on Dependency Tree Skeletons. In *Proceedings of the Workshop on Textual Entailment and Paraphrasing*, pages 36–41, Prague, June 2007.
7. Witten, I. H. and Frank, E. *Weka: Practical Machine Learning Tools and Techniques with Java Implementations*. Morgan Kaufmann, 1999.