DFKI–LT at AVE 2007: Using Recognizing Textual Entailment for Answer Validation

Rui Wang and Günter Neumann

LT-lab, DFKI
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
{wang.rui, neumann}@dfki.de

Abstract. This report is about our participation in the *Answer Validation Exercise* (AVE) 2007. Our system utilizes a *Recognizing Textual Entailment* (RTE) system as a component to validate answers. We first change the question and the answer into *Hypothesis* (**H**) and view the document as *Text* (**T**), in order to cast the AVE task into a RTE problem. Then, we use our RTE system to tell us whether the entailment relation holds between the documents (i.e. **T**s) and question-answer pairs (i.e. **H**s). Finally, we adapt the results for the AVE task. In all, we have submitted two runs and achieved f-measures of 0.46 and 0.55 respectively, which both outperform last year's best result for English. After detailed error analysis, we have found that both the recall and the precision of our system could be improved in the future.

Keywords: Answer Validation, Recognizing Textual Entailment

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 is to evaluate the answers obtained by the former stages of a QA system and select the most proper answers for the final output.

A lot of research has been done on this topic. In recent years, a new trend is to use Recognizing Textual Entailment (RTE-1 – Dagan et al., 2006; RTE-2 – Bar-Haim et al., 2006) to do answer validation, see 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 forms 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 fairly high 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, one of our motivations is to improve answer validation by using RTE, and the other is to test our RTE system in other NLP applications.

This report will start with introducing our AVE system, which consists of the preprocessing part, the RTE component, and the post-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

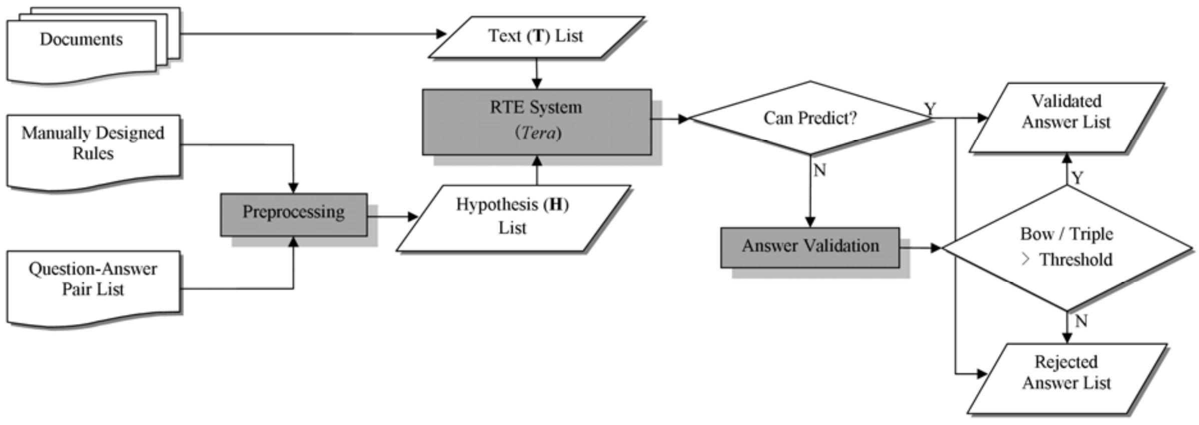

Figure 1 Architecture of our AVE system

Our AVE system uses our RTE system (*Tera* – Textual Entailment Recognition for Application) as a core component, and includes preprocessing and post-processing modules. The preprocessing module mainly adapts questions, their corresponding answers, and supporting documents into *Text*(**T**)-*Hypothesis*(**H**) pairs, assisted by some manually designed patterns. The post-processing module (i.e. the Answer Validation in Figure 1) will validate each answer and select a most proper one based on the output of the RTE system. We will see the details of each component in the coming sections.

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, the question is,

In which country was Edouard Balladur born? (id=178)[1]

The QA system gives out several candidate answers to this question, as follows,

Frances (id=178_1)
12% jobless rate (id=178_3)
7 (id=178_5)
...

Each answer will have one supporting document where the answer comes from, like this,

Paris, Wednesday CONSERVATIVE Prime Minister Edouard Balladur, defeated in France's presidential election, resigned today clearing the way for President-elect Jacques Chirac to form his own new government. Balladur's move was a formality since outgoing President Francois Mitterrand hands over power next week to Chirac, the conservative Paris mayor who won last Sunday's run-off election... (parts) (id=178_1)
…

The assumption here is that *if the answer is relevant to the question, the document which contains the answer should entail the statement derived by combining the question and the answer*. This section will mainly focus on the combination of the question and the answer and in the next sections the RTE system and how to deal with the output of the system will be described.

In order to combine the question and the answer into a statement, we need some language patterns. Normally, we have different types of questions, such as *Who*-questions asking about persons, *What*-questions asking about definitions, etc. Therefore, we manually construct some language patterns for the input questions. For the example given above (id=178), we will apply the following pattern,

Edouard Balladur was born in <Answer>. (id=178)

Consequently, we substitute the "<Answer>" by each candidate answer to form **H**s – hypotheses. Since the supporting documents are naturally the **T**s – texts, the **T**-**H** pairs are built up accordingly,

Id: *178_1*
Entailment: *Unknown*
Text: *Paris, Wednesday CONSERVATIVE Prime Minister* ***Edouard Balladur, defeated in France's*** *presidential election, resigned today clearing the way for President-elect Jacques Chirac to form his own new government… (parts)*

[1] All the examples in this report come from AVE 2007 test data, i.e. "AVE2007-annotated-test-EN.xml".

Hypothesis: *Edouard Balladur was born in* ***Frances****.*

These **T**-**H** pairs can be the input for any the generic RTE system.

2.2 The RTE Component

The RTE component is based on the RTE system we have used for RTE-3 Challenge (Giampiccolo et al., 2007). The system contains a main approach with two backup strategies. The main approach extracts parts of the dependency structures to form a new representation, named *Tree Skeleton*, as the feature space and then applies *Subsequence Kernels* to represent TSs and perform Machine Learning. 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**. We will begin with the main approach and briefly introduce the backup strategies at the end of this section.

If we take a broad view of the RTE task, in essence, we are asked to tell whether a particular relationship (i.e. entailment) holds between two text fragments. Notice that this kind of relationship is one-directional, which is from **T** to **H**. Generally, people start with **T**, do some processing, and then check whether **H** is reachable. However, we did it in the opposite direction, based on the observations: 1) **H** is the target we want to verify, which leads us to identify the relevant parts of **T**; and 2) **H** (i.e. a question and one of its candidate answers) is usually textually shorter than **T** (i.e. a document or a snippet). The **T**-**H** pair (id=178_1) is just an example of this. All the information we need in **T** is the part in bold of the first sentence.

Now the remaining problems are: 1) How do we identify the relevant parts of **T** based on **H**? 2) How do we combine them? 3) How do we represent them? The three steps of the main approach are aiming to solve these problems: extracting tree skeletons to obtain the most relevant parts, merging them to define the feature space, and applying subsequence kernels to represent the features and perform the machine learning procedure.

Tree Skeleton Extraction

Since tree skeletons are extracted based on the dependency structures, we need to use some dependency parsers to obtain the dependency parse trees. We have used Minipar (Lin, 1998). The following Figure 2 shows the output given the previous **H** as the input sentence. As well as **H** is usually textually shorter than **T**, the dependency structure of **H** is also simpler. From Figure 2, we can easily identify the structure of the whole sentence: There are two nouns in the lower part of the parse tree, and they share a common parent node, which is a verb in the upper part. Since content words usually convey most of the meaning of the sentence, we will mark the two nouns as *Topic Words* and the verb as the *Root Node*. Together with the dependency paths in between, they form a subtree of the original dependency structure, which can be viewed as an extended version of *Predicate-Argument Structure* (Gildea and Palmer, 2002). We call the subtree *Tree Skeleton*, the topic words *Foot Nodes*, and the dependency path from the noun to the root node *Spine*. If there are two foot nodes, the corresponding spines will be the *Left Spine* and the *Right Spine*.

On top of the tree skeleton of **H**, the tree skeleton of **T** can also be extracted. We assume that *if the entailment holds from* ***T*** *to* ***H****, at least, they will share the same topics*. Since in practice, there are different expressions for the same entity, we have applied some fuzzy matching techniques to correspond the topic words in **T** and **H**, like

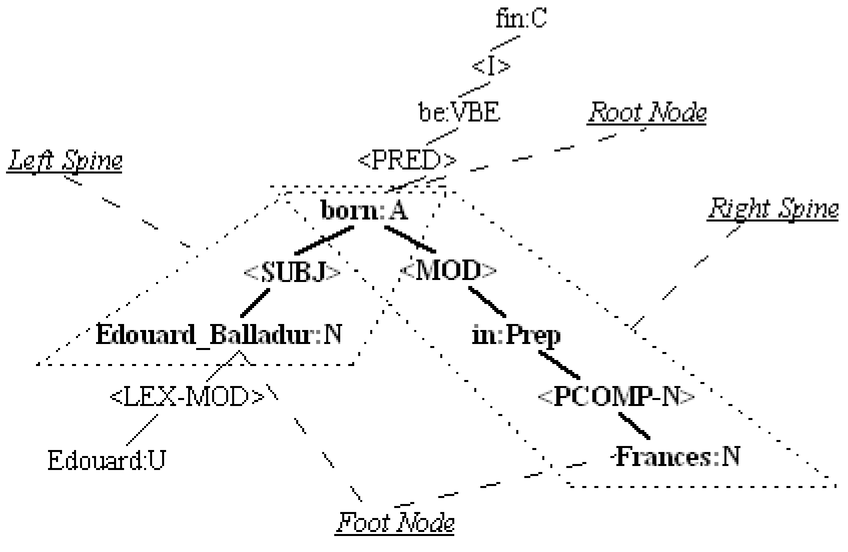

Figure 2 Dependency Structure and Tree Skeleton

initialism, partial matching, etc. Once we successfully identify the topic words in **T**, we trace up along the dependency parse tree to find the lowest common parent node, which will be marked as the root node of the tree skeleton of **T**[2].

Notice that the prerequisite for performing this method is: topic words are identified and corresponded between **T** and **H**. Furthermore, if there are only two topic words, the whole tree skeleton can be viewed as a sequence of words and dependency relation tags in a flat structure. This is important because sequence structures are much less complex than the tree structures, which will greatly reduce the computational complexity. In practice, 37% of the RTE-2 test data (Wang and Neumann, 2007a) and 36% of the RTE-3 test data (Wang and Neumann, 2007b) meet these requirements and consequently can be dealt with by the main approach.

Spine Generalization and Merging

Before moving on, some generalizations are necessary in order to avoid the scarcity of features in the data. Several steps will be performed: 1) We will collapse some of the dependency relation tags from the parsers to more generalized names, e.g., collapsing <OBJ2> and <DESC> to <OBJ>; 2) we will group together all nodes that have relation labels like <CONJ> or <NN>, since they are assumed to refer to the same entity or belong to one class of entities sharing some common characteristics; 3) lemmas are removed except for the topic words. Finally, we will get the generalized tree skeleton as follows (# is a separator to mark the root node),

Edouard_Balladur:N <SUBJ> #born:A# <MOD> PREP <PCOMP-N> Frances:N

We will do the same on the tree skeleton of **T**, and then merge the two tree skeletons by 1) excluding the longest common prefixes for left spines and 2) excluding the longest common suffixes for right spines. Finally, we will get the dissimilarity of the two tree skeletons and we call it *Spine Differences*, i.e. *Left Spine Difference* (LSD) and *Right Spine Difference* (RSD), like the following (## is to separate the parts from **T** and **H**, and *null* means an empty string),

LSD: *<OBJ> ## <SUBJ>*

RSD: *N ## (null)*

Subsequence Kernels Application

After the spine generalization and merging, now all the remaining symbols are POS tags and (generalized) dependency relation tags. They altogether form a *Closed-Class Symbol* (CCS) set. The spine difference is thus a sequence of CCSs. To represent it, we have utilized a Subsequence Kernel and a Collocation Kernel. The definitions of the two kernels are as below,

$$K_{subsequence}(<T,H>,<T',H'>) = \sum_{i=1}^{|T|}\sum_{i'=1}^{|T'|} K_{CCS}(CCS_i, CCS_{i'}) + \sum_{j=1}^{|H|}\sum_{j'=1}^{|H'|} K_{CCS}\big(CCS_j, CCS_{j'}\big)$$

$$K_{collocation}(<T,H>,<T',H'>) = \sum_{i=1}^{|T|}\sum_{i'=1}^{|T'|}\sum_{j=1}^{|H|}\sum_{j'=1}^{|H'|} K_{CCS}(CCS_i, CCS_{i'}) \cdot K_{CCS}\big(CCS_j, CCS_{j'}\big)$$

whereby *T* and *H* refers to all spine differences from **T** and **H**, and $|T|$ and $|H|$ represent the cardinalities. The function $K_{CCS}(CCS,CCS')$ checks whether its arguments are equal.

As well as these two kernels, we have also considered the comparison between root nodes and their adjacent dependency relations. We have observed that some adjacent dependency relations of the root node (e.g. <SUBJ> or <OBJ>) can play important roles in predicting the entailment relationship. For instance, the verb "*sell*" has a direction of the action from the subject to the object. In addition, the verb "*sell*" and "*buy*" convey totally different semantics. Therefore, we assign them two extra simple kernels named *Verb Consistence* (VC) and *Verb Relation Consistence* (VRC). The former indicates whether two root nodes have a similar meaning, and the latter indicates whether the relations are contradictive (e.g. <SUBJ> and <OBJ> are contradictive).

On top of all these four kernels, we have used a composite kernel to combine them linearly with different weights,

$$K_{composite} = \alpha K_{subsequence} + \beta K_{collocation} + \gamma K_{VC} + \delta K_{VRC}$$

where γ and δ are learned from the training corpus, and $\alpha=\beta=1$.

Backup Strategies

As well as the main approach, we have two backup strategies as well: one is called the *Triple Similarity* and the other is called the *BoW Similarity*.

Dependency structures can be represented in a form of a triple set, which expresses the local dependency relations. A triple is of the form <node1, relation, node2>, where node1 represents the head, node2 the modifier, and relation the dependency relation. Thus, each dependency parse tree consists of a set of such triples.

Chief requirements for the backup strategy are robustness and simplicity. Accordingly, we construct a similarity function, which operates on two triple sets and determines how many triples of **H** are contained in **T**.

[2] The Root Node of **T** is not necessary to be a verb, instead, it could be a noun, a preposition, or even a dependency relation.

The core assumption here is that *the higher the number of matching triple elements, the more similar both sets are, and the more likely it is that **T** entails **H***. The function uses an approximate matching function. Different cases (i.e. ignoring either the parent node or the child node, or the relation between nodes) might provide different indications for the similarity of **T** and **H**. In all cases, a successful match between two nodes means that they have the same lemma and POS. We then sum them up using different weights and divide the result by the cardinality of **H** for normalization. The different weights learned from the corpus indicate that the "amount of missing linguistic information" affect entailment decisions differently.

The BoW similarity score is calculated by dividing the number of overlapping words between **T** and **H** by the total number of words in **H** after a simple tokenization according to the space between words.

2.3 Post-processing

The RTE component has given us several things: 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; 3) every **T**-**H** pair has 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 threshold is learned from the training corpus and the two similarity scores are used in different submission runs.

For the adaption back to the AVE task, the "YES" entailment cases will be validated answers and the "NO" entailment cases will be rejected answers. 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

The AVE task this year asks the system to judge whether an answer extracted from a document is a valid answer to the given question. The result can be either "VALIDATED" or "REJECTED", which mean it's a valid answer or not respectively. Furthermore, among all the "VALIDATED" answers to each question, one best answer will be marked as "SELECTED", but if there is no "VALIDATED" answers, there will be no "SELECTED" answer, either.

The AVE training data contains 200 questions, 1121 answers and 1121 supporting documents, among which there are 130 validated answers and 991 rejected answers. The AVE testing data contains 67 questions, 202 answers and supporting documents, among which there are 21 validated answers, 174 rejected answers, and 7 unknown answers according to the gold standard.

We have submitted two runs. Both of the two runs we have used the main approach and one backup strategy. The difference is that in the first run, the BoW similarity score is the backup, while in the second run, the triple similarity score is taken. Our machine learning process is performed by using the classifier SMO from the WEKA toolkit (Witten and Frank, 1999). In the following, we will first show the table of the results and then present an error analysis.

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

Table 1 Results of our two runs

Though the absolute scores are not very promising, they are still better than all the results for English from last 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 for now. 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 does not appear in the documents. Instead, a pattern like "*Kiri Te 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. A truly difficult example is the following one,

> ***Question:*** *Which American President* ***masterminded the Camp David Agreement****?* (id=160)
> ***Answer:*** *Jimmy Carter.* (id=160_2)
> ***Document:*** *United States President Jimmy Carter invited both Sadat and Begin to a summit* ***at Camp David*** *to* ***negotiate*** *a final peace.*

Not only the lexical semantics of “*mastermind*” and “*negotiate*” are necessary, but also some world knowledge like the name of an agreement is usually the place where people subscribe it.

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. Errors from the parser will also cause problems. For example,

> ***Question:*** *Who is Thom Rotella?* (id=106)
> ***Answer:*** *Grant Geissman.* (id=106_3)
> ***Document:*** *As founder of Positive Music Records, Navarro is responsible for launching and furthering the recording careers of saxophonists Bob Militello and Brandon Fields,* **guitarists Grant Geissman, Thom Rotella and Pat Kelley***, and keyboardists Gregg Karukas and Marcus Johnson.*

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 participation of AVE 2007. The work presented is utilizing our RTE system to validate answers from QA systems. One the one hand, it is an effective way to improve 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 method.

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

Acknowledgement

This 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).

References

1. Bar-Haim, R., Dagan, I., Dolan, B., Ferro, L., Giampiccolo, D., Magnini, B. and Szpektor, I. 2006. The Second PASCAL Recognising Textual Entailment Challenge. In Proceedings of the Second PASCAL Challenges Workshop on Recognising Textual Entailment, Venice, Italy.
2. Bunescu, R. and Mooney, R. 2006. Subsequence Kernels for Relation Extraction. In Advances in Neural Information Processing Systems 18. MIT Press.
3. Dagan, I., Glickman, O., and Magnini, B. 2006. The PASCAL Recognising Textual Entailment Challenge. In Quiñonero-Candela et al., editors, MLCW 2005, LNAI Volume 3944, pages 177-190. Springer-Verlag.
4. Giampiccolo, D., Magnini, B., Dagan, I., and Dolan, B. 2007. The Third PASCAL Recognizing Textual Entailment Challenge. In Proceedings of the Workshop on Textual Entailment and Paraphrasing, pages 1–9, Prague, June 2007.
5. Gildea, D. and Palmer, M. 2002. The Necessity of Parsing for Predicate Argument Recognition. In Proceedings of the 40th Meeting of the Association for Computational Linguistics (ACL 2002):239-246, Philadelphia, PA.
6. Lin, D. 1998. Dependency-based Evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems.
7. Peñas, A., Rodrigo, Á., Sama, V., and Verdejo, F. 2006. Overview of the Answer Validation Exercise 2006. In the AVE 2006 Working Notes.
8. Wang, R. and Neumann, G. 2007a. Recognizing Textual Entailment Using a Subsequence Kernel Method. In Proc. of AAAI 2007.
9. 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.
10. Witten, I. H. and Frank, E. Weka: Practical Machine Learning Tools and Techniques with Java Implementations. Morgan Kaufmann, 1999.