Information Synthesis for Answer Validation

Rui Wang Jr[1] and Günter Neumann[2]

[1] Saarland University
66123 Saarbrücken, Germany
rwang@coli.uni-sb.de
[2] LT-Lab, DFKI
Stuhlsatzenhausweg 3, 66123 Saarbrücken, Germany
neumann@dfki.de

Abstract. This report is about our participation in the *Answer Validation Exercise* (AVE2008). Our system casts the AVE task into a *Recognizing Textual Entailment* (RTE) problem and uses an existing RTE system to validate answers. Additional information from *named-entity* (NE) recognizer, question analysis component, and so on, is also considered as assistances to make the final decision. In all, we have submitted two runs, one run for English and the other for German. They have achieved f-measures of 0.64 and 0.61 respectively. Compared with our system last year, which purely depends on the output of the RTE system, the extra information does show its effectiveness.

Keywords: Answer Validation, Recognizing Textual Entailment, Information Synthesis

1 Introduction and Related Work

Answer Validation is an important step for *Question Answering* (QA) systems, which aims to validate the answers extracted from natural language texts, and select the most proper answers for the final output.

Using *Recognizing Textual Entailment* (RTE-1 – Dagan et al., 2006; RTE-2 – Bar-Haim et al., 2006) to do answer validation has shown a great success (Peñas et al., 2007). We also developed our own RTE system and participated in AVE2007. The RTE system 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.

However, on the AVE data sets, we still found much space for the improvement. Therefore, based on the system we developed last year, our motivation this year is to see whether using extra information, e.g. *named-entity* (NE) recognition, question analysis, etc., can make further improvement on the final results.

This report will start with a brief introduction of our RTE system and then followed by the whole AVE system. The results of our two submission runs will be shown in section 4, and in the end, we will summarize our work.

2 The RTE System

The RTE system (Wang and Neumann, 2007a; Wang and Neumann, 2007b) is developed 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**.

The main approach starts with processing **H**, since it is usually textually shorter than **T**, and the dependency structure also simpler. Tree skeletons are extracted based on the dependency structures derived by Minipar (Lin, 1998) for English and SMES (Neumann and Piskorski, 2002) for German. There are nouns in the lower part of the parse tree, and they share a common parent node, which is (usually) a verb in the upper part. Since content words usually convey most of the meaning of the sentence, we will mark the 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 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**[1].

After some generalizations, we 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). Then, since 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* (Wang and Neumann, 2007a).

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).

Finally, the main approach is assisted by two backup strategies: one is called the *Triple Similarity* and the other is called the *BoW Similarity*. Chief requirements for the backup strategy are robustness and simplicity. Accordingly, we construct a similarity function, which operates on two triple (dependency structure represented in the form of <head, relation, modifier>) sets and determines how many triples of **H** are contained in **T**. 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**. We then sum them up using different weights and divide the result by the cardinality of **H** for normalization. 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.

[1] 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.

3 The AVE System

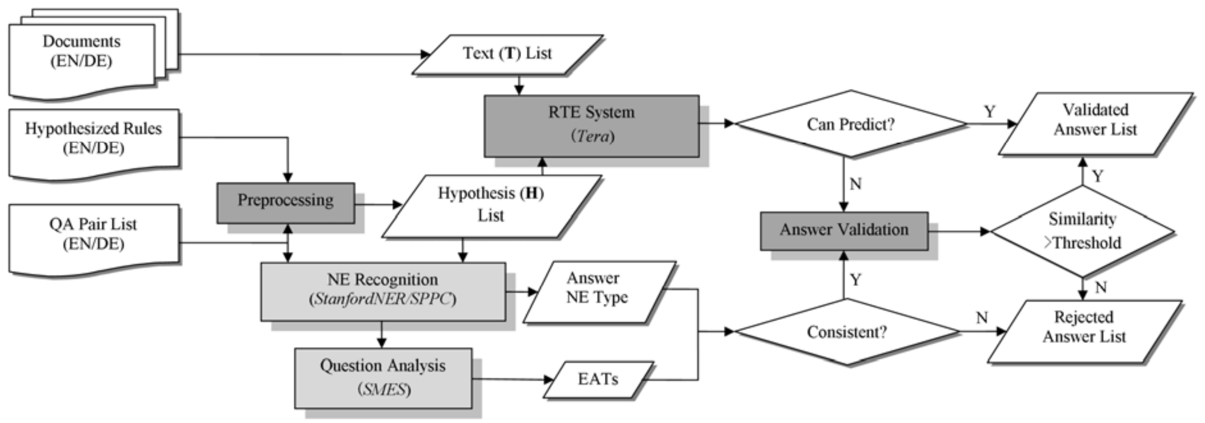

Fig. 1. Our AVE system uses the RTE system (**Tera** – *Textual Entailment Recognition for Application*) as a core component. The preprocessing module mainly adapts questions, their corresponding answers, and supporting documents into *Text* (**T**)-*Hypothesis* (**H**) pairs, assisted by some manually designed patterns. The post-processing module (i.e. the *Answer Validation* in the picture) will validate each answer and select a most proper one based on the output of the RTE system. The new modules added are the *NE Recognition* and *Question Analysis*. Thus, we will have extra information like NEs in the answers, *Expected Answer Types* (EATs), etc.

3.1 Preprocessing and Post-processing

Since the input of the AVE task is a list of questions, their corresponding answers and the documents containing these answers, we need to adapt them into **T**-**H** pairs for the RTE system. For instance, the question is,

How many "Superside" world championships did Steve Webster win between 1987 and 2004? (id=87)[2]

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

ten (id=87_1)
24 (id=87_2)
…

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

The most successful sidecar racer in Superside has been Steve Webster MBE, who has won ten world championships between 1987 and 2004. (id=87_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=87), we will apply the following pattern,

Steve Webster won <Answer> *"Superside" world championships between 1987 and 2004.* (id=87)

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: *87_1*
Entailment: *Unknown*
Text: *The most successful sidecar racer in Superside has been Steve Webster MBE, who has won ten world championships between 1987 and 2004.*
Hypothesis: *Steve Webster won* ***ten*** *"Superside" world championships between 1987 and 2004.*

These **T**-**H** pairs can be the input for any generic RTE systems. In practice, after applying our RTE system, if the **T**-**H** pairs are covered by our main approach, we will directly use the answers; if not, we will use a threshold

[2] The "id" comes from AVE 2008 test data, i.e. "AVE2008-annotated-test-EN.xml".

to decide the answer based on the two similarity scores. Therefore, every **T**-**H** pair has a triple similarity score and a BoW similarity score, and for some of the **T**-**H** pairs, we directly know whether the entailment holds. The post-processing is straigh tforward, the "YES" en tailment cases will be v alidated 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.

3.2 Additional Components

The RTE syst em is used as a core c omponent of the AVE system. Based on the error analysis of last year's results, this year we use additional components to filter out noisy candi dates. Therefore, two extra components are added to the architecture, the NE recognizer and the question analyzer. For NE recognition, we use StanfordNER (Finkel et al., 2005) for English and SPPC (Neumann and Piskorski, 2002) for German; and for question analysis, we use the SMES system (Neumann and Piskorski, 2002). The detailed workflow is as follows,

1. Annotate NEs in **H**, store them in an NE list; if the answer is an NE, store the NE type as A'_Type;
2. Analyze the question and obtain expected answer type, store it as A_Type;
3. Synthesize all the information, i.e. NE list, A_Type, A'_Type, BoW similarity, Triple similarity, etc.

As for the example mentioned above (id=87), the additional information will be,

NE list: *Steve Webster* (person), *1987* (date), *2004* (date);
A_Type: Number
A'_Type: Number

Then, heuristic rules are st raightforward to be applied, e.g. c hecking the consistence betwee n A_Type and A'_Type, checking whether all (or how many of) the NEs also appear in the documents, etc. All these results together with the outputs of the RTE system will be syn thesized to make the final decision.

4 Results

We have submitted two runs for this year's AVE tasks, one for English and one for German. In the following, we will first show the table of the results and then present an error analysis.

Table 1. Results of our submissions compared with last year's

Submission Runs	Recall	Precision	F-measure	Estimated QA Performance	QA Accuracy
100% VALIDATED (EN)	1	0.08	0.14	N/A	N/A
50%VALIDATED (EN)	0.5	0.08	0.13	N/A	N/A
Perfect Selection (EN)	N/A	N/A	N/A	0.56	**0.34**
Best QA System (EN)	N/A	N/A	N/A	0.21	**0.21**
dfki07-run1 (EN)	0.62	0.37	**0.46**	N/A	0.16
dfki07-run2 (EN)	0.71	0.44	**0.55**	N/A	0.21
dfki08run1 (EN)	0.78	0.54	**0.64**	0.34	**0.24**
100% VALIDATED(DE)	1	0.12	0.21	N/A	N/A
50% VALIDATED (DE)	0.5	0.12	0.19	N/A	N/A
Perfect Selection (DE)	N/A	N/A	N/A	0.77	**0.52**
Best QA System (DE)	N/A	N/A	N/A	0.38	**0.38**
dfki08run1 (DE)	0.71	0.54	0.61	0.52	**0.43**

In the table, we notice that both for Eng lish and German, our validation system outperforms the best QA systems, which suggests the necessity of the validation step. Although there is a gap between the system performance and the perfect selection, the results are quite satisfactory. If we compare this year's results with last year's, the additional information does improve the results significantly.

Comparing the recall and precision, for both languages, the latter is worse. Therefore, we did some error analysis to see whether there is still some space for improvements. An interesting example in the English data is as follows,

Question: *What is the name of the best known piece by Jeremiah Clarke?* (id=0011)
Answer: *a rondo* (id=0011_7)
Document: *The most famous piece known by that name, however, is a composition by Jeremiah Clarke, properly **a rondo** for keyboard named Prince of Denmark's March.*

Our system wrongly validated this answer, because "*a rondo*" is not the name of that music work. In fact, what we need here is a special proper name recognizer which can differentiate whether the noun is a name for a music work.

In the German data, other kinds of errors occur. For instance,

Question: *Wer war Russlands Verteidigungsminister **1994**?* (id=0020[3])
Answer: *Pawel Gratschow* (id=0020_6)
Document: *Wie der russische Verteidigungsminister Pawel Gratschow am Mittwoch in Tiflis weiter bekanntgab, will Rußland insgesamt fünf Militärstützpunkte in den Kaukasus-Republiken Georgien, Armenien und Aserbaidschan einrichten. **1994**-02-02*

The key problem here is that the year "*1994*" in the document might not be the year when the event happened, but the year of the report. This asks us to further synthesize the information we have, i.e. NE annotation and dependency parsing, to make better use of them.

5 Conclusion and Future Work

To sum up, in this paper, we described our participation of AVE 2008. Based on the experience of last year's participation, apart from the RTE core system, we add two extra components, NE recognizer and question analyzer, to further improve the results. The strategy is quite successful according to the comparison of system performances.

However, the problem has not been fully solved. Due to the noisy web data, filtering some documents in the preprocessing step could be even more effective than working on the post-processing phase. Another direction considered by us is to take a closer look at the different performances between different languages.

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. Jenny Rose Finkel, Trond Gr enager, and Christopher Manning. 2005. Incorporating Non-local Information into Information Extraction Systems by Gibbs Sampling. Proceedings of the 43nd Annual Meeting of the Association for Computational Linguistics (ACL 2005), pp. 363-370.
5. 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.
6. 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.
7. Lin, D. 1998. Dependency-based Evaluation of MINIPAR. In Workshop on the Evaluation of Parsing Systems.
8. Neumann, G. and Piskorski , J. 2002. A Shallow Text Processing Core Engine. Journal of Computational Intelligence, Volume 18, Number 3, 2002, pages 451-476.
9. Anselmo Peñas, Álvaro Rodrigo, Felisa Verdejo. 2007. Overview of the Answer Validation Exercise 2007. In the CLEF 2007 Working Notes.
10. Wang, R. and Neumann, G. 2007a. Recognizing Textual Entailment Using a Subsequence Kernel Method. In Proc. of AAAI 2007.
11. 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.
12. Wang, R. and Neumann, G. 2007c. DFKI–LT at AVE 2007: Using Recognizing Textual Entailment for Answer Validation. In online proceedings of CLEF 2007 Working Notes, ISBN: 2-912335-31-0, September 2007, Budapest, Hungary.

[3] This "id" comes from "AVE2008-annotated-test-DE.xml".