Combining Deterministic Dependency Parsing and Linear Classification for Robust RTE

Alexander Volokh alvo01@dfki.de
Günter Neumann neumann@dfki.de
Bogdan Sacaleanu bogdan@dfki.de

DFKI
Stuhlsatzenhausweg 3
66123 Saarbrücken, Germany

Abstract

We present a robust RTE approach which is built as one module incorporating all possible knowledge sources in form of different features. This way we can easily include or remove knowledge sources which are involved into the process of judging the entailment relation. We perform numerous tests in which we analyse the contribution of different types of features based on word forms, structural information, lexical semantics and named entity recognition to this process. The core of our system is our own deterministic dependency parser MDParser, which is based on a fast linear classification approach. We use the RTE6 challenge as an opportunity to evaluate its performance in a real-world application against another state of the art parser MaltParser. In our official submissions we achieve an f-score of 39.81 with MaltParser and 38.26 with MDParser. However, the parsing speed with MDParser is 26 times higher.

1 Introduction

Textual entailment is a relation between text fragments, which states whether the meaning of one fragment is contained in the other one. The entailing text fragment is usually called text (T), the entailed fragment is usually called hypothesis (H), and both are usually referred to as T/H pair. Being able to recognise this relation can significantly support many NLP applications in numerous fields, such as IE – finding different text variants that entail the same target relation, QA – finding texts, which entail the expected answer, IR – retrieved documents entail the query, or summarisation – the summary is entailed by the original text (Dagan, Glickman and Magnini, 2006). Another particularly interesting and novel application of textual entailment has been tried out in the Semeval-2 workshop (task #12 - PETE) in order to evaluate parsers independently of the grammar formalism they are based on (Yuret, Han and Turgut, 2010).

Our team has been actively participating in the RTE challenges organised over the past several years. However, since the conditions of the challenges, as well as the used data vary every year, our approaches and systems also undergo significant changes from year to year.

In the beginnings of RTE the data sets were rather small and the number of important phenomena required to deal with was rather limited. Thus we had tried to identify the most important phenomena and to develop a solution to deal with the fragments involving these phenomena. The resulting systems consisted then of a collection of problem-specific modules, which were used depending on the input and of a fallback strategy, which was used in case no problem-specific solution existed for the given data. The decision about which module to use for which problem, usually consisted of a rule-based voting mechanism (cf. Wang and Neumann, 2007a; Wang and Neumann 2007b).

Starting from the last year, however, the amount of data has grown significantly. On the one hand the task became more challenging, since the increased data size diversified the linguistic problems one has to deal with. On the other hand it became possible to develop robust systems, which are able to deal with RTE in more realistic conditions, since the larger data collections required the systems to be more general and not

tuned to the particular data. Additionally, the larger amounts of data allowed to apply machine learning techniques more efficiently. In particular, the voting mechanism could be automatised, since it became possible to learn which source of information should be used for judging textual entailment and to what extent.

For last year's task our team has tried to extract both syntactic and semantic information from text fragments (Wang, Zhang and Neumann, 2009). As the source for syntactic information a dependency analysis of the text fragments was used. For semantic information a number of components, including semantic role labelling, and lexical semantics resources, such as VerbOcean and WordNet, as well as a coreference resolver, were used. As in previous years the system consisted of a sophisticated component, which tried to classify each T-H-pair and of a simple fallback component in case the module was not applicable.

In this year's task the focus lied on the exploration of contribution of different linguistic components to the overall result of an RTE system. Therefore, and for the reasons of robustness, we have designed one single component (no main and fallback strategies as before), which incorporates all possible knowledge sources, which one desires to include. This way it is very easy to add/remove one of the knowledge sources and to measure its contribution. The voting mechanism, automated or not, whether to apply a main strategy or a fallback strategy is no longer a factor. Since we were very fond of the PETE shared task, we have decided to also use RTE6 as an opportunity to evaluate the role of parsers in RTE. However, for time reasons we were able to compare only two parsers: our own dependency parser MDParser[1] and a well-known state of the art parser MaltParser (Nivre et al., 2006).

We have focused our work on the main task of this year's challenge. Even though we have submitted two runs for the Novelty Detection task as well, and achieved 78.87 F-measure as primary score and 36.76 as justification score with MDParser, and 79.26 f-score as primary score and 36.43 as justification score with MaltParser, the approach was not adopted to this task in any way. That is why we will restrict ourselves exclusively to the Main task in this paper.

[1] http://mdparser.sb.dfki.de/

This year we have tried out a feature-based approach in order to control which linguistic components are used for judging the entailment relation. In Section 2 we describe our parser, which is the core of our RTE system. In Section 3 we introduce the representation which we produce for all T-H-pairs. In Section 4 we explain how we classify whether H is entailed by T. In Section 5 we list all the features we have used in our models and in Section 6 describe how the models are trained. In Section 7 we explain our ablation tests, which were mandatory for all participants. In Section 8 we present the comparison between MDParser and MaltParser. Finally, we discuss and summarise our results in Section 9.

2 MDParser

MDParser stands for multilingual dependency parser and is a data-driven system, which can be used to parse text of an arbitrary language for which training data is available. It is a transition-based parser and uses a deterministic version of the Covington's algorithm (Covington, 2000).

The models of the system are based on various features, which are extracted from the words of the sentence, including word forms and part of speech tags. No additional morphological features or lemmas are currently used in our models, even if they are available in the training data, since the system is especially designed for processing plain text in different languages, and such components are not available for every language.

The preprocessing components of MDParser include a.) a sentence splitter[2], since the parser constructs a dependency structure for individual sentences, b.) a tokenizer, in order to recognise the elements between which the dependency relations will be built[3], and c.) a part of speech tagger, in order to determine the part of speech tags, which are intensively used in the feature models[4].

MDParser is an especially fast system because it uses a linear classification algorithm L1R-LR(L1 regularised logistic regression) from the

[2] http://morphadorner.northwestern.edu/morphadorner/sentencesplitter/

[3] http://morphadorner.northwestern.edu/morphadorner/wordtokenizer/

[4] The part of speech tagger was trained with the SVMTool http://www.lsi.upc.edu/~nlp/SVMTool/

machine learning package LibLinear (Lin et al., 2008) for constructing its dependency structures and therefore it is particularly suitable for processing very large amounts of data. Thus it can be used as a part of larger applications in which dependency structures are desired.

Additionally, significant efforts were made in order to make the gap between our linear classification and more advanced methods as small as possible, e.g. by introducing features conjunctions, which are complex features built out of ordinary features, as well as methods for automatically measuring feature usefulness in order to automate and optimise feature engineering.

3 Triple Representation

Every parser usually produces its own somehow special representation of the sentence. We have created such a representation, which we will call *triple representation* and have implemented an automatic transformation of the results of MaltParser, and of course MDParser into it. We have also managed to transform the results of Stanford Parser into this format, as well as to adopt MDParser's dependencies to the same annotation format[5], but could not manage to compare our RTE systems with this parser before the deadline.

The triple representation of a sentence is a set of triple elements of the form <*parent*, *label*, *child*>, where *child* and *parent* elements stand for the head and the modifier words and their parts of speech, and *label* stands for the relation between them. E.g. <have:VBZ, SBJ, Somebody:NN>. Moreover each triple contains the indexes for the child and parent words in order to be able to construct a dependency tree out of the set of triples, if necessary. A full triple could thus look as follows: <triple parentIndex="2" childIndex="1"> have:VBZ, SBJ, Somebody:NN</triple>.

This information is extractable from the results of any dependency parser.

4 Predicting Entailment

Given a corpus, a hypothesis H, and a set of "candidate" sentences retrieved by Lucene from that corpus for H, the RTE system was required to identify all the sentences that entail H. We have constructed triple representations for all candidate sentences (T's) and hypotheses (H's) and have formulated a large set of feature templates[6] which are applied to each pair of T and H in order to measure the similarity of both sentences and judge the entailment relation. The feature templates intended to capture:

a) how similar are the dependency structures of T and H

b) how similar are the words and their semantics in T and H

c) how similar are the named entities in T and H.

We will now describe each of these important aspects of our comparison in more details.

Structural Similarity

In our former work (Volokh and Neumann, 2010) we have found out that in order to compare the dependency structures of two sentences it is sensible to analyse and compare only the top-level structure of the dependency trees rather than the complete trees, because

a) the most important pieces of information are usually encoded as the root of the sentence and/or its arguments rather than somewhere deeply embedded.

b) the chances that the dependency parsers will produce absolutely accurate results up to the depth of 3 or more is rather low, so its more robust to restrict oneself to the top-level dependency relations which are also the ones which can be usually recognised with the least effort

Word-level Similarity

Very often the triples will not perfectly match because the word forms used may vary. In this case it is important to investigate whether the words used for expressing certain relations are at least semantically related, if they do not have the same word forms.

Therefore we have used two similarity measures based on WordNet: Jcn (Jiang and Conrath, 1997) and Lin (Lin, 1998). Both of them are implemented in the package we have used - JWNL (Java WordNet Library). Additionally this package contains an English dictionary and methods for looking up lemmas for any word forms.

Named Entities

[5]Stanford Parser uses stanford dependencies, whereas MaltParser (typically) uses CoNLL-X dependencies. MDParser can produce structures of both types.

[6]The full set of used feature templates will be presented in Section 5

Named entities are a special case of nouns, which often occur in very important positions and for which the measurement of semantic similarity is at least as important as for other content words. However, the method based on WordNet, which we have described above will not work for named entities since they are not part of the dictionary.

Therefore we have used the LingPipe Named Entity Recogniser (Alias-i, 2008) in order to be able to tell, whether a named entity occurring in a certain position in H and a different named entity occurring in the same position in T are at least of the same type (PERSON, LOCATION; ORGANIZATION; DATE, NUMBER).

5 Feature Model

In this section we describe the features used in our system:

1. Similarity of root triples of T and H
   I) same roots
   II) different roots
2. For each dependent triples of the root of H examine whether such triple(a triple with the same *label*) is also present in T
   I) dependent is not present in T
   II) dependent is present in T
   IIa) and both *child* and *parent* are the same
   IIb) and the *child* is the same but the *parent* is different
   IIc) and the *parent* is different but the *child* is the same
   IId) but both the child and the parent are different
3. For each dependent of the dependents of the root of H (depth 2 in the dependency tree) examine the corresponding triple in T. The values are computed the same way as in 2.
4. For all triple-pairs which are being compared according to 2 or 3, the following feature templates are used:
   a) the word form of the H-triple' child is taken
   b) the word form of the H-triple's parent is taken
   c) the word form of the T-triple's child is taken
   d) the word form of the T-triple's parent is taken
   e) the POS-tag of the H-triple' child is taken
   f) the POS-tag of the H-triple's parent is taken
   g) the POS-tag of the T-triple's child is taken
   h) the POS-tag of the T-triple's parent is taken
5. For all triple-pairs which are being compared according to 2 or 3, the following feature templates are used:
   a) The similarity of H-triple's child and T-triple's child is measured with JCn and according to it the following values are used:
   I) JCn similarity is < 0.5, else
   II) JCn similarity is < 1, else
   III) JCn similarity is < 1.5, else
   IV) JCn similarity is < 2, else
   V) JCn similarity is < 2.5, else
   VI) otherwise
   b) The similarity of H-triple's child and T-triple's child is measured with Lin and according to it the following values are used:
   I) Lin similarity is < 0.2, else
   II) Lin similarity is < 0.4, else
   III) Lin similarity is < 0.6, else
   IV) Lin similarity is < 0.8, else
   V) Lin similarity is < 1, else
   VI) otherwise(actually the only case is Lin=1)
6. The percentage of the verbs and nouns occurring in H also occurring in T is computed:
   I) Less than 20% of verbs and nouns occurring in H are present in T, else
   II) Less than 40%, else
   III) Less than 60%,
   IV) Less than 80%
   V) 100%
7. The percentage of the named entities occurring in H also occurring in T is computed:
   I) Less than 20% of named entities occurring in H are present in T, else
   II) Less than 40%, else
   III) Less than 60%,
   IV) Less than 80%
   V) 100%

8. For every type of named entities (PERSON, LOCATION, ORGANIZATION) in H check whether entities of the same type are also present in T:
   I) There are entities of the same type
   II) There are no entities of the same type

6 Classification

We use the same machine learning approach as we have used for training the models for our parser and train models based on the features described in the previous section. Thus we get a classifier able to distinguish between "YES" and "NO" candidates. For the training we have used all available candidate sentences. However, among those overall 15955 T-H-pairs only 897 belong to the category "YES" and the other almost 95% of the candidates get the label "NO". This highly imbalanced training set causes the classifier to tend to classify most of the candidates as "NO".

However, since the classification method we have used is a probabilistic one, the classification result is not simply one of the two possible classes, namely "YES" or "NO", but rather a probability distribution over these classes. Thus we could decide to classify a candidate as "YES" not only when the probability of this event was above 50%, but we were free to define a threshold and to treat a T-H-pair as entailed already when the probability of the "YES" class was above 13%, 15%, 17% or any other value.

In our official submission we have used 17% as the threshold. In two of our ablation runs we have varied this value in order to test its influence on the overall performance.

7 Results

The result of our official submission with MDParser was: precision = "53.31", recall = "29.84", F-measure = "38.26". The results of our submissions, where MaltParser was used in order to construct the triple representations, was: precision = "55.94" recall = "30.90" F-measure = "39.81". This score is ranked 7th out of 18 participating systems. The result of MaltParser is thus 1.55 higher than the one of MDParser. However, we will treat the system with MDParser as our main system and our ablation tests, which we will present later, are based on this system.

We have also submitted some runs for the Novelty Detection task, which ranked 4th out of 9 (F-measure = "78.87"). For these runs exactly the same system, i.e. without absolutely any adaptations to the task, was used.

8 Parser Comparison

Our whole approach is based on the comparison of triple representations of T and H. Thus the approach is highly dependent on the quality of these structures, which are constructed by a dependency parser.

The results presented in this paper are based on our own dependency parser – MDParser, which was shortly introduced in the section 2. MDParser uses a very fast machine learning technique and usually achieves slightly inferior results (using UAS/LAS evaluation metrics on standard test data, e.g. CoNLL(cf. Buchholz and Marsi, 2006)) in comparison to other parsers, which for example use kernel-based classification or other more sophisticated methods. One of our motivations was to evaluate the performance of our parser not only on the standard test data sets, but also in a concrete real life application.

We have already known from our former experience, e.g. from participating in the SemEval-2 Task #12 (Volokh and Neumann, 2010), that despite the straightforwardness of our system, in real life applications, where only the most important dependency relations have to be identified correctly and not the complete structure, the difference in accuracy between our system and other more sophisticated systems fades and the difference in parsing speed remains.

We have used the RTE6 task as another opportunity in order to evaluate the performance of our parser in comparison to more sophisticated systems. However, due to time reasons we have only managed to compare MDParser with MaltParser. For a comparison between MDParser and StanfrodParser we required a different model, which operates with Stanford dependencies and not with CoNLL dependencies and we have managed to achieve that only after the submission deadline.

As far as parsing times are concerned it takes 73188ms to parse 3376 different sentences, consisting overall of 74326 tokens, from the RTE6

test data set with MDParser and 1954684ms to parse the same sentences with MaltParser (parsing time averaged over 3 runs, since it varied slightly from run to run).

The following table summarises the figures of the comparison:

	Parsing Time	Sentences per Second	Tokens per Second
MDParser	73.188s	46.128	1015.55
MaltParser	1954.684s	1.73	38.02

The result of MDParser is thus 1.5 points worse in terms of F-measure, but can be achieved 26 times faster.

The results described above are based on the MaltParser model we have trained ourselves. We have used the following options for training:

LibSVM options: "-s_0_-t_1_-d_2_-g_0.2_-c_1.0_-r_0.4_-e_0"
root_handling: strict
parsing_algorithm: nivrestandard
data_split_column: POSTAG
data_split_threshold: 100

We are aware that especially the option *root_handling=strict* made the system slower than it would be with *root_handling=normal*, however during the development phase the accuracy of our approach was much higher with the triple representation constructed with the strict root_handling, so we have stayed with these options.

9 Ablation Tests

We have performed 10 ablation tests in order to examine how much the different types of features presented in Section 5 contributed to the overall performance of the system and whether the inclusion of some of them was a mistake, since a better result could otherwise be achieved.

Here is the overview of the test with a short description which types of features were left out in each of the tests (the numbers correspond to the Section 5 feature description).

Test #	F Measure	Impact	Left out Features
1	35.31	2.95	1 (root features)
2	39.11	-0.85	2 (depth 1 features)
3	38.54	-0.28	3 (depth 2 features)
4	33.27	4.99	4 (word form and pos features)
5	19.22	19.04	6 (content word features)
6	36.04	2.22	7 (named entities features)
7	38.49	-0.23	5 (WordNet similarity features)
8	36.72	1.54	Coreference resolution features. No additional features were introduced or left out, but the content of all T-H-pairs was first processed with the LingPipe coreference resolution tool.
9	39.10	-0.84	Threshold 0.15
10	39.11	-0.85	Threshold 0.13

From these results we can infer that our models were far from optimal. On the one hand some features were apparently harmful and the system could have performed better without them. On the other hand some features, e.g. coreference resolution were not included, which turned out to be a mistake, since they would have had a positive impact. Last of all, the threshold for classifying a candidate as entailed was not chosen optimally neither.

Otherwise we see, that all levels of comparison between the text of T and the text of H turned out to be useful. The most important step is to analyse the similarity of the texts on the word level (abl. tests 5 and 7); structural comparison (test 1-4) and named entity comparison(test 6) improve the performance further to a slightly smaller degree.

10 Conclusions

We present a robust feature-based approach in order to judge the entailment relation. The architecture of our system allows us to easily include or remove knowledge sources and measure their contribution to the overall result. We perform numerous tests in order to measure the influence of

different linguistic components on the overall result. Most of the features are extracted from the output of a dependency parser, which is the core of our system. We use a particularly fast deterministic dependency parser MDParser, which is based on linear classifiers in order to produce the dependency structures for the RTE data. In order to compare the performance of our parser we also try out our approach with a different state of the art dependency parser MaltParser. The MaltParser is slightly better for the accuracy of our system but is significantly slower.

Acknowledgements

The work presented here was partially supported by a research grant from the German Federal Ministry of Economics and Technology (BMWi) to the DFKI project Theseus Ordo TechWatch (FKZ: 01MQ07016). We thank Joakim Nivre and Johan Hall for their support and tips when training models with MaltParser. Additionally, we are very grateful to Sven Schmeier for providing us with a trained part of speech tagger for English and for his support when using this tool.

References

Michael A. Covington, 2000. *A Fundamental Algorithm for Dependency Parsing*. In Proceedings of the 39th Annual ACM Southeast Conference.

Dan Klein and Christopher D. Manning, 2003. *Accurate Unlexicalized Parsing*. Proceedings of the 41st Meeting of the Association for Computational Linguistics, pp. 423-430.

Lin D, 2003. *Dependency-Based Evaluation Of Minipar*. In Building and using Parsed Corpora Edited by: Abeillé A. Dordrecht: Kluwer; 2003.

Sabine Buchholz and Erwin Marsi. 2006. *CoNLL-X shared task on multilingual dependency parsing*. In Proceedings of CONLL-X, pages 149–164, New York.

Ido Dagan, Oren Glickman and Bernardo Magnini. *The PASCAL Recognising Textual Entailment Challenge*. In Quinonero-Candela, J.; Dagan, I.; Magnini, B.; d'Alche-Buc, F. (Eds.), Machine Learning Challenges. Lecture Notes in Computer Science, Vol. 3944, pp. 177-190, Springer, 2006.

Nivre, J., J. Hall and J. Nilsson, 2006. *MaltParser: A Data-Driven Parser-Generator for Dependency Parsing*. In Proceedings of the fifth international conference on Language Resources and Evaluation (LREC2006), pp. 2216-2219, May 24-26, 2006, Genoa, Italy.

Alexander Volokh and Günter Neumann, 2010. *372:Comparing the Benefit of Different Dependency Parsers for Textual Entailment Using Syntactic Constraints Only.* In Proceedings of the SemEval-2010 Evaluation Exercises on Semantic Evaluation.

Rui Wang and Günter Neumann, 2007. *Recognizing textual entailment using sentence similarity based on dependency tree skeletons*. In Proceedings of the ACL-PASCAL Workshop on Textual Entailment and Paraphrasing, pp. 36-41, Prague, Czech Republic.

Rui Wang and Günter Neumann, 2007. *Recognizing Textual Entailment Using a Subsequence Kernel Method*. In Proceedings of AAAI 2007.

R. Fan, K. Chang, C. Hsieh, X. Wang, and C. Lin, 2008. *LIBLINEAR: A Library for Large Linear Classification*. Journal of Machine Learning Research, 9(4): 1871–1874.

Rui Wang, Yi Zhang, and Günter Neumann. 2009. A *Joint Syntactic-Semantic Representation for Recognizing Textual Relatedness*. In Text Analysis Conference TAC 2009 WORKSHOP Notebook Papers and Results, Pages 1-7, National Institute of Standards and Technology (NIST), Gaithersburg, Maryland, USA

Deniz Yuret, Aydın Han and Zehra Turgut, 2010. *SemEval-2010 Task 12: Parser Evaluation using Textual Entailments.* In Proceedings of the SemEval-2010 Evaluation Exercises on Semantic Evaluation.

Maltparser. http://maltparser.org/

MDParser: Multilingual Dependency Parser. http://mdparser.sb.dfki.de/

Alias-i. 2008. LingPipe 4.0.0. http://alias-i.com/lingpipe