MARS: A Specialized RTE System for Parser Evaluation

Rui Wang[†] **Yi Zhang**[†‡]
† Department of Computational Linguistics, Saarland University
‡ LT-Lab, German Research Center for Artificial Intelligence
Im Stadtwald, 66123 Saarbrücken, Germany
{rwang,yzhang}@coli.uni-sb.de

Abstract

This paper describes our participation in the the SemEval-2010 Task #12, *Parser Evaluation using Textual Entailment*. Our system incorporated two dependency parsers, one semantic role labeler, and a deep parser based on hand-crafted grammars. The shortest path algorithm is applied on the graph representation of the parser outputs. Then, different types of features are extracted and the entailment recognition is casted into a machine-learning-based classification task. The best setting of the system achieves 66.78% of accuracy, which ranks the 3rd place.

1 Introduction

The SemEval-2010 Task #12, *Parser Evaluation using Textual Entailment* (PETE) (Yuret et al., 2010), is an interesting task connecting two areas of research, parsing and recognizing textual entailment (RTE) (Dagan et al., 2005). The former is usually concerned with syntactic analysis in specific linguistic frameworks, while the latter is believed to involve more semantic aspects of the human languages. However, no clear-cut boundary can be drawn between syntax and semantics for both tasks. In recent years, the parsing community has been reaching beyond what was usually accepted as syntactic structures. Many deep linguistic frameworks allow the construction of semantic representations in parallel to the syntactic structure. Meanwhile, data-driven shallow semantic parsers (or semantic role labelers) are another popular type of extension to enrich the information in the parser outputs.

Although *entailment* is described as a semantic relation, RTE, in practice, covers linguistic phenomena at various levels, from surface text to the meaning, even to the context and discourse. One proposal of solving the problem is to deal with different cases of entailment using different specialized RTE modules (Wang and Neumann, 2009). Then, the PETE data can be naturally classified into the syntactic and shallow semantic categories.

By participating in this shared task, we aim to investigate whether different parsing outputs leads to different RTE accuracy, and on the contrary, whether the "application"-based evaluation provides insights to the parser comparison. Further, we investigate if strict grammaticality checking with a linguistic grammar is helpful in this task.

2 System Description

The workflow of the system is shown in Figure 1 and the details of the three components will be elaborated on in the following sections.

2.1 Preprocessing

In this paper, we generally refer all the linguistic analyses on the text as *preprocessing*. The output of this procedure is a graph representation, which approximates the meaning of the input text. In particular, after tokenization and POS tagging, we did dependency parsing and semantic role labeling. In addition, HPSG parsing is a filter for ungrammatical hypotheses.

Tokenization and POS Tagging We use the Penn Treebank style tokenization throughout the various processing stages. **TnT**, an HMM-based POS tagger trained with Wall Street Journal sections of the PTB, was used to automatically predict the part-of-speech of each token in the texts and hypotheses.

Dependency Parsing For obtaining the syntactic dependencies, we use two dependency parsers, MSTParser (McDonald et al., 2005) and MaltParser (Nivre et al., 2007). MSTParser is a graph-based dependency parser where the best parse tree is acquired by searching for a spanning tree

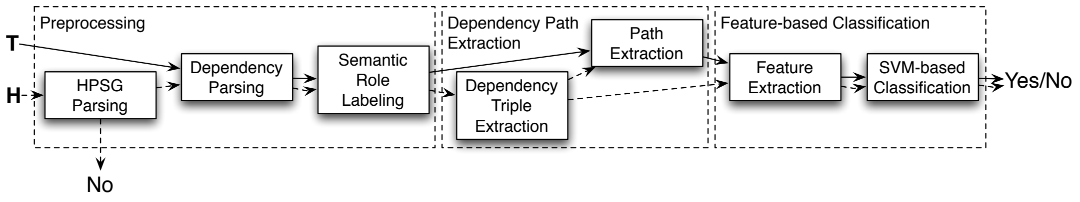

Figure 1: Workflow of the System

which maximize the score on an either partially or fully connected dependency graph. MaltParser is a transition-based incremental dependency parser, which is language-independent and data-driven. It contains a deterministic algorithm, which can be viewed as a variant of the basic shift-reduce algorithm. Both parsers can achieve state-of-the-art performance and Figure 2 shows the resulting syntactic dependency trees of the following T-H pair,

> ID: *2036*; Entailment: *YES*
> **T**: *Devotees of the market question the value of the work national service would perform.*
> **H**: *Value is questioned.*

Semantic Role Labeling The statistical dependency parsers provide shallow syntactic analyses of the entailment pairs through the limited vocabulary of the dependency relations. In our case, the CoNLL shared task dataset from 2008 were used to train the statistical dependency parsing models. While such dependencies capture interesting syntactic relations, when compared to the parsing systems with deeper representations, the contained information is not as detailed. To compensate for this, we used a shallow semantic parser to predict the semantic role relations in the **T** and **H** of entailment pairs. The shallow semantic parser was also trained with CoNLL 2008 shared task dataset, with semantic roles extracted from the Propbank and Nombank annotations (Zhang et al., 2008). Figure 3 shows the resulting semantic dependency graphs of the T-H pair.

HPSG Parsing We employ the English Resource Grammar (Flickinger, 2000), a hand-written linguistic grammar in the framework of HPSG, and the PET HPSG parser (Callmeier, 2001) to check the grammaticality of each hypothesis sentence. As the hypotheses in this PETE shared task were automatically generated, some ungrammatical hypotheses occur in non-entailment pairs. the grammaticality checking allows us to quickly identify these instances.

2.2 Dependency Path Extraction

According to the task definition, we need to verify whether those dependency relations in H also appear in T. We firstly find out all the important dependency triples in H, like <word, dependency relation, word>, excluding those having stop words. The extracted syntactic dependency triples of the example T-H pair would be none, since the only content words "value" and "questioned" have no direct syntactic dependency in-between (Figure 2). The extracted semantic dependency triples would be <"questioned", "A1", "value"> (Figure 3).

After that, we use the word pairs contained in the extracted dependency triples as anchors to find out the corresponding dependency relations in T. Notice that it is not necessarily that we can always find a direct dependency relation in T between the same word pair, so we need to traverse the dependency tree or graph to find the *dependency paths*. In general, we treat all the dependency trees and graphs as undirected graphs with loops, but keep records for the directions of the edges we traverse. For the following three representations, we apply slightly different algorithms to find the dependency path between two words,

Syntactic Dependency Tree We simply traverse the tree and find the corresponding dependency path connecting the two words;

Semantic Dependency Graph We apply Dijkstra's algorithm (Dijkstra, 1959) to find the shortest path between the two words;

Joint Dependency Graph We assign different weights to syntactic and semantic dependencies and apply Dijkstra's algorithm to find the shortest path (with the lowest cost)[1].

2.3 Feature-based Classification

Based on the meaning representation we have discussed above (Section 2.1 and Section 2.2), we ex-

[1]In practice, we simply give semantic dependencies 0.5 cost and syntactic dependencies 1.0 cost, to show the preferences on the former when both exist.

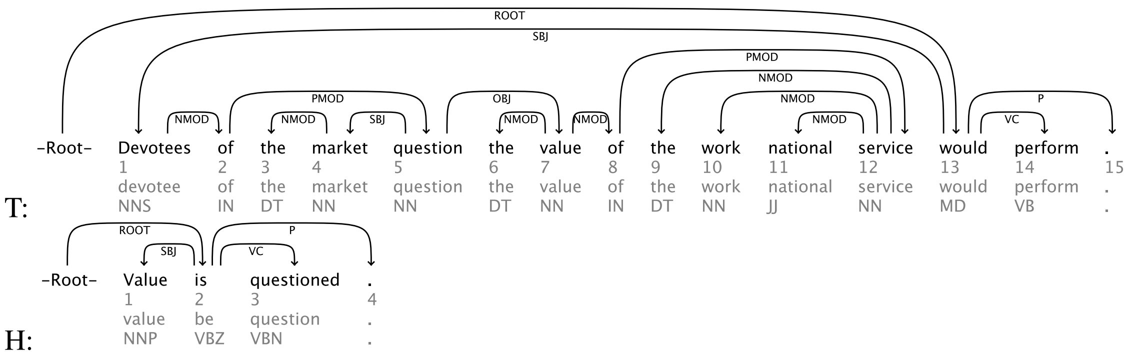
ROOT
SBJ
PMOD
NMOD
OBJ
VC
P
-Root- Devotees of the market question the value of the work national service would perform .
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
devotee of the market question the value of the work national service would perform .
NNS IN DT NN NN DT NN IN DT NN JJ NN MD VB .
T:
-Root- Value is questioned .
1 2 3 4
value be question .
NNP VBZ VBN .
H:

Figure 2: Syntactic dependency of the example T-H pair by MaltParser.

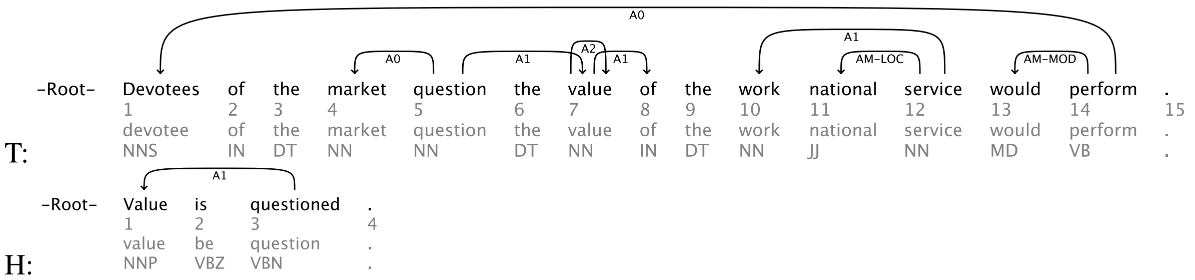
A0
A1
A2
AM-LOC
AM-MOD
-Root- Devotees of the market question the value of the work national service would perform .
1 2 3 4 5 6 7 8 9 10 11 12 13 14 15
devotee of the market question the value of the work national service would perform .
NNS IN DT NN NN DT NN IN DT NN JJ NN MD VB .
T:
-Root- Value is questioned .
1 2 3 4
value be question .
NNP VBZ VBN .
H:

Figure 3: Semantic dependency of the example T-H pair by MaltParser and our SRL system.

tract features for the machine-learning-based classifier. First of all, we should check whether there are dependency triples extracted from H, otherwise for our system, there is no meaning representation for that sentence. Then we also need to check whether the same words can be found in T as well. Only if the corresponding dependency paths are successfully located in T, we could extract the following features.

The direction of each dependency relation or path could be interesting. The direction of the H-path is clear, so we only need to check the direction of the T-path. In practice, we simply use a boolean value to represent whether T-path contains dependency relations with different directions. For instance, in Figure 3, if we extract the path from "market" to "value", the directions of the dependency relations contained in the path would be $\leftarrow$ and $\rightarrow$, one of which would be inconsistent with the dependency relation in H.

Notice that all the dependency paths from H have length 1[2], but the lengths of the dependency paths from T are varied. If the latter length is also 1, we can simply compare the two dependency relations; otherwise, we compare each of the dependency relation contained the T-path with H-path one by one[3]. By comparison, we mainly focus on two values, the category of the dependency relation (e.g. syntactic dependency vs. semantic dependency) and the content of the dependency relation (e.g. A1 vs. AM-LOC).

We also incorporate the string value of the dependency relation pair and make it boolean according to whether it occurs or not. Table 1 shows the feature types we extract from each T-H pair.

3 Experiments

As we mentioned in the preprocessing section (Section 2.1), we utilize the open source dependency parsers, MSTParser[4] and MaltParser[5], our own semantic role labeler (Zhang et al., 2008), and the PET HPSG parser[6]. For the shortest path algorithm, we use the jGraphT package[7]; and for the machine learning toolkit, we use the UniverSVM

[2]The length of one dependency path is defined as the number of dependency relations contained in the path.

[3]Enlightened by Wang and Neumann (2007), we exclude some dependency relations like "CONJ", "COORD", "APPO", etc., heuristically, since in most of the cases, they will not change the relationship between the two words at both ends of the path.

[4]`http://sourceforge.net/projects/mstparser/`

[5]`http://maltparser.org/`

[6]`http://heartofgold.dfki.de/PET.html`

[7]`http://jgrapht.sourceforge.net/`

	H_Null?	T_Null?	Dir	Multi?	Dep_Same?	Rel_Sim?	Rel_Same?	Rel_Pair
Joint	+	+	+	+	+	+	+	+
No Sem		+	+	+			+	+
No Syn	+	+	+	+		+	+	+

Table 1: Feature types of different settings of the system. *H_Null?* means whether H has dependencies; *T_Null?* means whether T has the corresponding paths (using the same word pairs found in H); *Dir* is whether the direction of the path T the same as H; *Multi?* adds a prefix, *m_*, to the *Rel_Pair* features, if the T-path is longer than one dependency relation; *Dep_Same?* checks whether the two dependency types are the same, i.e. syntactic and semantic dependencies; *Rel_Sim?* only occurs when two semantic dependencies are compared, meaning whether they have the same prefixes, e.g. *C-*, *AM-*, etc.; *Rel_Same?* checks whether the two dependency relations are the same; and *Rel_Pair* simple concatenates the two relation labels together. Notice that, the first seven feature types all contain boolean values, and for the last one, we make it boolean as well, by observing whether that pair of dependency labels appear or not.

package[8]. We test different dependency graphs and feature sets as mentioned before (Table 1), and the results are shown in Table 2.

	MSTParser+SRL			MaltParser+SRL		
	Joint	No Sem	No Syn	Joint	No Sem	No Syn
+GC	0.5249	0.5116 (-1.3%)	0.5050 (-2.0%)	0.6678	0.5282 (-14.0%)	0.6346 (-3.3%)
-GC	0.5216	0.5050	0.4950	0.6545	0.5282	0.6179

Table 2: Experiment results of our system with different settings.

First of all, in almost all the cases, the grammaticality checking based on HPSG parsing is helpful, if we compare each pair of results at the two rows, +GC and -GC. In all cases, the joint graph representation achieves better results. This indicates that features extracted from both syntactic dependency and shallow semantic dependency are useful for the entailment recognition. For the MaltParser case, the semantic features show great importance. Notice that the performance of the whole system does not necessarily reflect the performance of the parser itself, since it also depends on our entailment modules. In all, the best setting of our system ranks the 3rd place in the evaluation.

4 Conclusion

In this paper, we present our system used in the PETE task, which consists of preprocessing, dependency path extraction, and feature-based classification. We use MSTParser and MaltParser as dependency parsers, our SRL system as a shallow semantic parser, and a deep parser based on hand-crafted grammars for grammaticality checking. The entailment recognition is done by an SVM-based classifier using features extracted from the graph representation of the parser outputs. Based on the results, we tentatively conclude that both the syntactic and the shallow semantic features are useful. A detailed error analysis would be our on-going work in the near future.

[8]`http://www.kyb.mpg.de/bs/people/fabee/universvm.html`

Acknowledgment

The authors thank the PIRE PhD scholarship and the German Excellence Cluster of MMCI for the support of the work.

References

Ulrich Callmeier. 2001. Efficient parsing with large-scale unification grammars. Master's thesis, Universität des Saarlandes, Saarbrücken, Germany.

Ido Dagan, Oren Glickman, and Bernardo Magnini. 2005. The pascal recognising textual entailment challenge. In Quiñonero-Candela et al., editor, *MLCW 2005*, volume LNAI Volume 3944, pages 177–190. Springer-Verlag.

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

Dan Flickinger. 2000. On building a more efficient grammar by exploiting types. *Natural Language Engineering*, 6(1):15–28.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005. Non-Projective Dependency Parsing using Spanning Tree Algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530, Vancouver, Canada.

Joakim Nivre, Jens Nilsson, Johan Hall, Atanas Chanev, Gülsen Eryigit, Sandra Kübler, Svetoslav Marinov, and Erwin Marsi. 2007. Maltparser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(1):1–41.

Rui Wang and Günter Neumann. 2007. Recognizing textual entailment using a subsequence kernel method. In *Proceedings of AAAI-07*, Vancouver, Canada, July.

Rui Wang and Günter Neumann. 2009. An accuracy-oriented divide-and-conquer strategy for recognizing textual entailment. In *Proceedings of TAC 2008*, 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*.

Yi Zhang, Rui Wang, and Hans Uszkoreit. 2008. Hybrid learning of dependency structures from heterogeneous linguistic resources. In *Proceedings of the Twelfth Conference on Computational Natural Language Learning (CoNLL 2008)*, pages 198–202, Manchester, UK.