SemEval-2010 Task 12: Parser Evaluation using Textual Entailments

Deniz Yuret
Koç University
İstanbul, Turkey
dyuret@ku.edu.tr

Aydın Han
Koç University
İstanbul, Turkey
ahan@ku.edu.tr

Zehra Turgut
Koç University
İstanbul, Turkey
zturgut@ku.edu.tr

Abstract

Parser Evaluation using Textual Entailments (PETE) is a shared task in the SemEval-2010 Evaluation Exercises on Semantic Evaluation. The task involves recognizing textual entailments based on syntactic information alone. PETE introduces a new parser evaluation scheme that is formalism independent, less prone to annotation error, and focused on semantically relevant distinctions.

1 Introduction

Parser Evaluation using Textual Entailments (PETE) is a shared task that involves recognizing textual entailments based on syntactic information alone. Given two text fragments called "text" and "hypothesis", textual entailment recognition is the task of determining whether the meaning of the hypothesis is entailed (can be inferred) from the text. In contrast with general RTE tasks (Dagan et al., 2009) the PETE task focuses on syntactic entailments:

Text: The man with the hat was tired.
Hypothesis-1: The man was tired. *(yes)*
Hypothesis-2: The hat was tired. *(no)*

PETE is an evaluation scheme based on a natural human linguistic competence (i.e. the ability to comprehend sentences and answer simple yes/no questions about them). We believe systems should try to model natural human linguistic competence rather than their dubious competence in artificial tagging tasks.

The PARSEVAL measures introduced nearly two decades ago (Black et al., 1991) still dominate the field of parser evaluation. These methods compare phrase-structure bracketings produced by the parser with bracketings in the annotated corpus, or "treebank". Parser evaluation using short textual entailments has the following advantages compared to treebank based evaluation.

Consistency: Recognizing syntactic entailments is a more natural task for people than treebank annotation. Focusing on a natural human competence makes it practical to collect high quality evaluation data from untrained annotators. The PETE dataset was annotated by untrained Amazon Mechanical Turk workers at an insignificant cost and each annotation is based on the unanimous agreement of at least three workers. In contrast, of the 36306 constituent strings that appear multiple times in the Penn Treebank (Marcus et al., 1994), 5646 (15%) have multiple conflicting annotations. If indicative of the general level of inconsistency, 15% is a very high number given that the state of the art parsers claim f-scores above 90% (Charniak and Johnson, 2005).

Relevance: PETE automatically focuses attention on semantically relevant phenomena rather than differences in annotation style or linguistic convention. Whether a phrase is tagged ADJP vs ADVP rarely affects semantic interpretation. Attaching the wrong subject to a verb or the wrong prepositional phrase to a noun changes the meaning of the sentence. Standard treebank based evaluation metrics do not distinguish between semantically relevant and irrelevant errors (Bonnema et al., 1997). In PETE semantically relevant differences lead to different entailments, semantically irrelevant differences do not.

Framework independence: Entailment recognition is a formalism independent task. A common evaluation method for parsers that do not use the Penn Treebank formalism is to automatically convert the Penn Treebank to the appropriate formalism and to perform treebank based evaluation (Nivre et al., 2007a; Hockenmaier and Steedman,

Proceedings of the 5th International Workshop on Semantic Evaluation, ACL 2010, pages 51–56,
Uppsala, Sweden, 15-16 July 2010.

2007). The inevitable conversion errors compound the already mentioned problems of treebank based evaluation. In addition, manually designed treebanks do not naturally lend themselves to unsupervised parser evaluation. Unlike treebank based evaluation, PETE can compare phrase structure parsers, dependency parsers, unsupervised parsers and other approaches on an equal footing.

PETE was inspired by earlier work on representations of grammatical dependency, proposed for ease of use by end users and suitable for parser evaluation. These include the grammatical relations (GR) by (Carroll et al., 1999), the PARC representation (King et al., 2003), and Stanford typed dependencies (SD) (De Marneffe et al., 2006) (See (Bos and others, 2008) for other proposals). Each use a set of binary relations between words in a sentence as the primary unit of representation. They share some common motivations: usability by people who are not (computational) linguists and suitability for relation extraction applications. Here is an example sentence and its SD representation (De Marneffe and Manning, 2008):

Bell, based in Los Angeles, makes and distributes electronic, computer and building products.

nsubj(makes-8, Bell-1)
nsubj(distributes-10, Bell-1)
partmod(Bell-1, based-3)
nn(Angeles-6, Los-5)
prep-in(based-3, Angeles-6)
conj-and(makes-8, distributes-10)
amod(products-16, electronic-11)
conj-and(electronic-11, computer-13)
amod(products-16, computer-13)
conj-and(electronic-11, building-15)
amod(products-16, building-15)
dobj(makes-8, products-16)

PETE goes one step further by translating most of these dependencies into natural language entailments.

Bell makes something.
Bell distributes something.
Someone is based in Los Angeles.
Someone makes products.

PETE has some advantages over representations based on grammatical relations. For example SD defines 55 relations organized in a hierarchy, and it may be non-trivial for a non-linguist to understand the difference between *ccomp* (clausal complement with internal subject) and *xcomp* (clausal complement with external subject) or between *nsubj* (nominal subject) and *xsubj* (controlling subject). In fact it could be argued that proposals like SD replace one artificial annotation formalism with another and no two such proposals agree on the ideal set of binary relations to use. In contrast, untrained annotators have no difficulty unanimously agreeing on the validity of most PETE type entailments.

However there are also significant challenges associated with an evaluation scheme like PETE. It is not always clear how to convert certain relations into grammatical hypothesis sentences without including most of the original sentence in the hypothesis. Including too much of the sentence in the hypothesis would increase the chances of getting the right answer with the wrong parse. Grammatical hypothesis sentences are especially difficult to construct when a (negative) entailment is based on a bad parse of the sentence. Introducing dummy words like "someone" or "something" alleviates part of the problem but does not help in the case of clausal complements. In summary, PETE makes the annotation phase more practical and consistent but shifts the difficulty to the entailment creation phase.

PETE gets closer to an extrinsic evaluation by focusing on semantically relevant, application oriented differences that can be expressed in natural language sentences. This makes the evaluation procedure indirect: a parser developer has to write an extension that can handle entailment questions. However, given the simplicity of the entailments, the complexity of such an extension is comparable to one that extracts grammatical relations.

The balance of what is being evaluated is also important. A treebank based evaluation scheme may mix semantically relevant and irrelevant mistakes, but at least it covers every sentence at a uniform level of detail. In this evaluation, we focused on sentences and relations where state of the art parsers disagree. We hope this methodology will uncover weaknesses that the next generation systems can focus on.

The remaining sections will go into more detail about these challenges and the solutions we have chosen to implement. Section 2 explains the method followed to create the PETE dataset. Sec-

tion 3 evaluates the baseline systems the task organizers created by implementing simple entailment extensions for several state of the art parsers. Section 4 presents the participating systems, their methods and results. Section 5 summarizes our contribution.

2 Dataset

To generate the entailments for the PETE task we followed the following three steps:

1. Identify syntactic dependencies that are challenging to state of the art parsers.
2. Construct short entailment sentences that paraphrase those dependencies.
3. Identify the subset of the entailments with high inter-annotator agreement.

2.1 Identifying Challenging Dependencies

To identify syntactic dependencies that are challenging for current state of the art parsers, we used example sentences from the following sources:

- The "Unbounded Dependency Corpus" (Rimell et al., 2009). An unbounded dependency construction contains a word or phrase which appears to have been moved, while being interpreted in the position of the resulting "gap". An unlimited number of clause boundaries may intervene between the moved element and the gap (hence "unbounded").
- A list of sentences from the Penn Treebank on which the Charniak parser (Charniak and Johnson, 2005) performs poorly[1].
- The Brown section of the Penn Treebank.

We tested a number of parsers (both phrase structure and dependency) on these sentences and identified the differences in their output. We took sentences where at least one of the parsers gave a different answer than the others or the gold parse. Some of these differences reflected linguistic convention rather than semantic disagreement (e.g. representation of coordination) and some did not represent meaningful differences that can be expressed with entailments (e.g. labeling a phrase ADJP vs ADVP). The remaining differences typically reflected genuine semantic disagreements that would effect downstream applications. These were chosen to turn into entailments in the next step.

[1] http://www.cs.brown.edu/~ec/papers/badPars.txt.gz

2.2 Constructing Entailments

We tried to make the entailments as targeted as possible by building them around two content words that are syntactically related. When the two content words were not sufficient to construct a grammatical sentence we used one of the following techniques:

- Complete the mandatory elements using the words "somebody" or "something". (e.g. To test the subject-verb dependency in "John kissed Mary." we construct the entailment "John kissed somebody.")
- Make a passive sentence to avoid using a spurious subject. (e.g. To test the verb-object dependency in "John kissed Mary." we construct the entailment "Mary was kissed.")
- Make a copular sentence or use existential "there" to express noun modification. (e.g. To test the noun-modifier dependency in "The big red boat sank." we construct the entailment "The boat was big." or "There was a big boat.")

2.3 Filtering Entailments

To identify the entailments that are clear to human judgement we used the following procedure:

1. Each entailment was tagged by 5 untrained annotators from the Amazon Mechanical Turk crowdsourcing service.
2. The results from the annotators whose agreement with the gold parse fell below 70% were eliminated.
3. The entailments for which there was unanimous agreement of at least 3 annotators were kept.

The instructions for the annotators were brief and targeted people with no linguistic background:

> Computers try to understand long sentences by dividing them into a set of short facts. You will help judge whether the computer extracted the right facts from a given set of 25 English sentences. Each of the following examples consists of a sentence (T), and a short statement (H) derived from this sentence by a computer. Please

> read both of them carefully and choose "Yes" if the meaning of (H) can be inferred from the meaning of (T). Here is an example:
>
> (T) *Any lingering suspicion that this was a trick Al Budd had thought up was dispelled.*
> (H) *The suspicion was dispelled.* Answer: YES
> (H) *The suspicion was a trick.* Answer: NO
>
> You can choose the third option "Not sure" when the (H) statement is unrelated, unclear, ungrammatical or confusing in any other manner.

The "Not sure" answers were grouped with the "No" answers during evaluation. Approximately 50% of the original entailments were retained after the inter-annotator agreement filtering.

2.4 Dataset statistics

The final dataset contained 367 entailments which were randomly divided into a 66 sentence development test and a 301 sentence test set. 52% of the entailments in the test set were positive.

Approximately half of the final entailments were from the Unbounded Dependency Corpus, a third were from the Brown section of the Penn Treebank, and the remaining were from the Charniak sentences. Table 1 lists the most frequent grammatical relations encountered in the entailments.

GR	Entailments
Direct object	42%
Nominal subject	33%
Reduced relative clause	21%
Relative clause	13%
Passive nominal subject	6%
Object of preposition	5%
Prepositional modifier	4%
Conjunct	2%
Adverbial modifier	2%
Free relative	2%

Table 1: Most frequent grammatical relations encountered in the entailments.

3 Baselines

In order to establish baseline results for this task, we built an entailment decision system for CoNLL format dependency files and tested several publicly available parsers. The parsers used were the Berkeley Parser (Petrov and Klein, 2007), Charniak Parser (Charniak and Johnson, 2005), Collins Parser (Collins, 2003), Malt Parser (Nivre et al., 2007b), MSTParser (McDonald et al., 2005) and Stanford Parser (Klein and Manning, 2003). Each parser was trained on sections 02-21 of the WSJ section of Penn Treebank. Outputs of phrase structure parsers were automatically annotated with function tags using Blaheta's function tagger (Blaheta and Charniak, 2000) and converted to the dependency structure with LTH Constituent-to-Dependency Conversion Tool (Johansson and Nugues, 2007).

To decide the entailments both the test and hypothesis sentences were parsed. All the content words in the hypothesis sentence were determined by using part-of-speech tags and dependency relations. After applying some heuristics such as active-passive conversion, the extracted dependency path between the content words was searched in the dependency graph of the test sentence. In this search process, same relation types for the direct relations between the content word pairs and isomorphic subgraphs in the test and hypothesis sentences were required for the "YES" answer.

Table 2 lists the baseline results achieved. There are significant differences in the entailment accuracies of systems that have comparable unlabeled attachment scores. One potential reason for this difference is the composition of the PETE dataset which emphasizes challenging syntactic constructions that some parsers may be better at. Another reason is the complete indifference of treebank based measures like UAS to the semantic significance of various dependencies and their impact on potential applications.

System	PETE	UAS
Berkeley Parser	68.1%	91.2
Stanford Parser	66.1%	90.2
Malt Parser	65.5%	89.8
Charniak Parser	64.5%	93.2
Collins Parser	63.5%	91.6
MST Parser	59.8%	92.0

Table 2: Baseline systems: The second column gives the performance on the PETE test set, the third column gives the unlabeled attachment score on section 23 of the Penn Treebank.

4 Systems

There were 20 systems from 7 teams participating in the PETE task. Table 3 gives the percentage of correct answers for each system. 12 sys-

System	Accuracy	Precision	Recall	F1
360-418-Cambridge	0.7243	0.7967	0.6282	0.7025
459-505-SCHWA	0.7043	0.6831	0.8013	0.7375
473-568-MARS-3	0.6678	0.6591	0.7436	0.6988
372-404-MDParser	0.6545	0.7407	0.5128	0.6061
372-509-MaltParser	0.6512	0.7429	0.5000	0.5977
473-582-MARS-5	0.6346	0.6278	0.7244	0.6726
166-415-JU-CSE-TASK12-2	0.5781	0.5714	0.7436	0.6462
166-370-JU-CSE-TASK12	0.5482	0.5820	0.4551	0.5108
390-433-Berkeley Parser Based	0.5415	0.5425	0.7372	0.6250
473-566-MARS-1	0.5282	0.5547	0.4551	0.5108
473-569-MARS-4	0.5249	0.5419	0.5385	0.5402
390-431-Brown Parser Based	0.5216	0.5349	0.5897	0.5610
473-567-MARS-2	0.5116	0.5328	0.4679	0.4983
363-450-VENSES	0.5083	0.5220	0.6090	0.5621
473-583-MARS-6	0.5050	0.5207	0.5641	0.5415
390-432-Brown Reranker Parser Based	0.5017	0.5217	0.4615	0.4898
390-435-Berkeley Parser with substates	0.5017	0.5395	0.2628	0.3534
390-434-Berkeley Parser with Self Training	0.4983	0.5248	0.3397	0.4125
390-437-Combined	0.4850	0.5050	0.3269	0.3969
390-436-Berkeley Parser with Viterbi Decoding	0.4784	0.4964	0.4359	0.4642

Table 3: Participating systems and their scores. The system identifier consists of the participant ID, system ID, and the system name given by the participant. Accuracy gives the percentage of correct entailments. Precision, Recall and F1 are calculated for positive entailments.

tems performed above the "always yes" baseline of 51.83%.

Most systems started the entailment decision process by extracting syntactic dependencies, grammatical relations, or predicates by parsing the text and hypothesis sentences. Several submissions, including the top two scoring systems used the C&C Parser (Clark and Curran, 2007) which is based on Combinatory Categorical Grammar (CCG) formalism. Others used dependency structures produced by Malt Parser (Nivre et al., 2007b), MSTParser (McDonald et al., 2005) and Stanford Parser (Klein and Manning, 2003).

After the parsing step, the decision for the entailment was based on the comparison of relations, predicates, or dependency paths between the text and the hypothesis. Most systems relied on heuristic methods of comparison. A notable exception is the MARS-3 system which used an SVM-based classifier to decide on the entailment using dependency path features.

Table 4 lists the frequency of various grammatical relations in the instances where the top system made mistakes. A comparison with Table 1 shows the direct objects and reduced relative clauses to be the frequent causes of error.

GR	Entailments
Direct object	51%
Reduced relative clause	36%
Nominal subject	20%
Object of preposition	7%
Passive nominal subject	7%

Table 4: Frequency of grammatical relations in entailment instances that got wrong answers from the Cambridge system.

5 Contributions

We introduced PETE, a new method for parser evaluation using textual entailments. By basing the entailments on dependencies that current state of the art parsers disagree on, we hoped to create a dataset that would focus attention on the long tail of parsing problems that do not get sufficient attention using common evaluation metrics. By further restricting ourselves to differences that can be expressed by natural language entailments, we hoped to focus on semantically relevant decisions rather than accidents of convention which get mixed up in common evaluation metrics. We chose to rely on untrained annotators on a natural inference task rather than trained annotators on an artificial tagging task because we believe (i) many subfields of computational linguistics are struggling to make progress because of the noise in artificially tagged data, and (ii) systems should try to model natural human linguistic competence rather than their dubious competence in artificial tagging tasks. Our hope is datasets like PETE will be used not only for evaluation but also for training and fine-tuning of systems in the future. Further

work is needed to automate the entailment generation process and to balance the composition of syntactic phenomena covered in a PETE dataset.

Acknowledgments

We would like to thank Laura Rimell, Stephan Oepen and Anna Mac for their careful analysis and valuable suggestions. Önder Eker contributed to the early development of the PETE task.

References

E. Black, S. Abney, D. Flickenger, C. Gdaniec, R. Grishman, P. Harrison, D. Hindle, R. Ingria, F. Jelinek, J. Klavans, et al. 1991. A Procedure for Quantitatively Comparing the Syntactic Coverage of English Grammars. In *Speech and natural language: proceedings of a workshop, held at Pacific Grove, California, February 19-22, 1991*, page 306. Morgan Kaufmann Pub.

D. Blaheta and E. Charniak. 2000. Assigning function tags to parsed text. In *Proceedings of the 1st North American chapter of the Association for Computational Linguistics conference*, page 240. Morgan Kaufmann Publishers Inc.

R. Bonnema, R. Bod, and R. Scha. 1997. A DOP model for semantic interpretation. In *Proceedings of the eighth conference on European chapter of the Association for Computational Linguistics*, pages 159–167. Association for Computational Linguistics.

Johan Bos et al., editors. 2008. *Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation*. In connection with the 22nd International Conference on Computational Linguistics.

J. Carroll, G. Minnen, and T. Briscoe. 1999. Corpus annotation for parser evaluation. In *Proceedings of the EACL workshop on Linguistically Interpreted Corpora (LINC)*.

E. Charniak and M. Johnson. 2005. Coarse-to-fine n-best parsing and MaxEnt discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, page 180. Association for Computational Linguistics.

S. Clark and J.R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics*, 33(4):493–552.

M. Collins. 2003. Head-driven statistical models for natural language parsing. *Computational linguistics*, 29(4):589–637.

I. Dagan, B. Dolan, B. Magnini, and D. Roth. 2009. Recognizing textual entailment: Rational, evaluation and approaches. *Natural Language Engineering*, 15(04).

M.C. De Marneffe and C.D. Manning, 2008. *Stanford typed dependencies manual*.

M.C. De Marneffe, B. MacCartney, and C.D. Manning. 2006. Generating typed dependency parses from phrase structure parses. In *LREC 2006*.

J. Hockenmaier and M. Steedman. 2007. CCGbank: a corpus of CCG derivations and dependency structures extracted from the Penn Treebank. *Computational Linguistics*, 33(3):355–396.

R. Johansson and P. Nugues. 2007. Extended constituent-to-dependency conversion for English. In *Proc. of the 16th Nordic Conference on Computational Linguistics (NODALIDA)*.

T.H. King, R. Crouch, S. Riezler, M. Dalrymple, and R. Kaplan. 2003. The PARC 700 dependency bank. In *Proceedings of the EACL03: 4th International Workshop on Linguistically Interpreted Corpora (LINC-03)*, pages 1–8.

D. Klein and C.D. Manning. 2003. Accurate unlexicalized parsing. In *Proceedings of the 41st Annual Meeting on Association for Computational Linguistics-Volume 1*, pages 423–430. Association for Computational Linguistics.

M.P. Marcus, B. Santorini, and M.A. Marcinkiewicz. 1994. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

R. McDonald, F. Pereira, K. Ribarov, and J. Hajic. 2005. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT/EMNLP*, pages 523–530.

J. Nivre, J. Hall, S. Kübler, R. McDonald, J. Nilsson, S. Riedel, and D. Yuret. 2007a. The CoNLL 2007 shared task on dependency parsing. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL*, volume 7, pages 915–932.

J. Nivre, J. Hall, J. Nilsson, A. Chanev, G. Eryigit, S. Kübler, S. Marinov, and E. Marsi. 2007b. MaltParser: A language-independent system for data-driven dependency parsing. *Natural Language Engineering*, 13(02):95–135.

S. Petrov and D. Klein. 2007. Improved inference for unlexicalized parsing. In *Proceedings of NAACL HLT 2007*, pages 404–411.

L. Rimell, S. Clark, and M. Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*, pages 813–821. Association for Computational Linguistics.