Language Resources and Evaluation manuscript No.
(will be inserted by the editor)

Parser Evaluation Using Textual Entailments

Deniz Yuret · Laura Rimell · Aydın Han

Received: date / Accepted: date

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. This paper describes the PETE task, gives an error analysis of the top-performing Cambridge system, and introduces a standard entailment module that can be used with any parser that outputs Stanford typed dependencies.

Keywords Parsing · Textual Entailments

1 Introduction

Parser Evaluation using Textual Entailments (PETE) is a shared task in the SemEval-2010 Evaluation Exercises on Semantic Evaluation that involves recognizing textual entailments to evaluate parser performance. Given two text

Deniz Yuret and Aydın Han
Koç University
Rumelifeneri Yolu
34450 Sarıyer, İstanbul, Turkey
Tel.: +90-212-338-1724
Fax: +90-212-338-1548
E-mail: dyuret,ahan@ku.edu.tr

Laura Rimell
Computer Laboratory
William Gates Building
15 JJ Thomson Avenue
Cambridge CB3 0FD, UK
Tel.: +44 (0)1223 334696
E-mail: laura.rimell@cl.cam.ac.uk

fragments called "text" (T) and "hypothesis" (H), recognizing textual entailment (RTE) is the task of determining whether the meaning of the hypothesis is entailed (can be inferred) from the text. In contrast to general RTE tasks (Dagan et al 2009) PETE is a *targeted textual entailment* task that 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)*

By syntactic entailments we mean entailments that can be recognized using grammatical knowledge alone, without recourse to background knowledge or logical reasoning. The main goal of PETE is not to create a general entailment system, but to use entailments as a probe to evaluate a basic linguistic competence, in this case identification of grammatical relations.

The participants were provided with a number of text – hypothesis sentence pairs as input (similar to the Text–Hypothesis-1 pair given above). The goal of the participating systems was to output an accurate YES/NO decision on the syntactic entailment status of each pair (e.g. YES for the Text–Hypothesis-1 pair and NO for the Text–Hypothesis-2 pair). Each entailment was focused on the relationship of a content word pair (e.g. man–tired for Hypothesis-1 and hat–tired for Hypothesis-2), however these content word pairs were not made available during testing. Table 1 provides some text–hypothesis examples from the actual test set. Section 2 provides further details on the dataset and Section 3 describes the participating systems and their results. All task relevant data is available at `http://pete.yuret.com`.

Text	**Hypothesis**	**Ent.**
There's a man with a wooden leg named Smith.	The **man** is **named** Smith.	YES
There's a man with a wooden leg named Smith.	The **leg** is **named** Smith.	NO
Two share a house almost devoid of furniture.	A **house** is **shared**.	YES
They wanted to touch the mystery.	They **wanted** the **mystery**.	NO
It took me five hours to write it that way.	Something **took hours**.	YES
Add things as you find you need'em.	The **things find** something.	NO

Table 1 Some text–hypothesis pair examples from the PETE test set. The third column gives the entailment status and the relevant content words are marked in bold.

1.1 Motivation

The motivation behind using targeted textual entailments as a test of linguistic competence is to use non-expert, native speaker judgements and achieve high inter-annotator agreement. It is generally difficult to achieve high inter-annotator agreement in artificial tagging tasks. We cite two examples: Dickinson and Meurers (2003) have found that of the 34,564 constituent strings that appear multiple times in Penn Treebank (Marcus et al 1994), 5,584 (16%)

have multiple conflicting annotations, of which an estimated 3,934 are errors. If indicative of the general level of inconsistency, 16% is a very high number given that state of the art parsers claim F-scores above 90% (Charniak and Johnson 2005). In the field of word sense disambiguation, Snyder and Palmer (2004), the organizers of the "English all-words task" in Senseval-3, state: "it seems that the best systems have hit a wall in the 65-70% range. This is not surprising given the typical inter-annotator agreement of 70-75% for this task."

The reason for the low inter-annotator agreement is usually not the annotators' lack of comprehension of the example sentences. The problem is their less than perfect ability to be consistent with the annotation guidelines, or the difficulty of coming up with consistent guidelines in the first place: The Penn Treebank annotation guidelines exceed 400 pages. WordNet (used in Senseval-3) defines more than 100,000 word senses, some of which are difficult to distinguish even for professional lexicographers. We believe the situation might be improved if our systems can target the natural competence of the annotators in comprehending sentences rather than their imperfect performance on artificial annotation tasks.

One can envision annotation tasks that probe the ability to identify grammatical relations, word senses, co-references etc. using basic sentence comprehension and generation skills rather than relying on detailed annotation guidelines or sense inventories. One example that targets the natural competence of annotators in comprehending and generating sentences is the lexical substitution task (McCarthy and Navigli 2007) introduced in SemEval-2007. Unlike standard word sense disambiguation tasks where the annotators need to comprehend not only the example sentences but the dictionary definitions of the target words, the lexical substitution task asks them to come up with substitutes for the target word that preserve the meaning of the sentence. Another example is (Erk et al 2009), which introduces a usage similarity task asking the annotators to judge the similarity between different usages of a word without relying on a sense inventory. Targeted textual entailment tasks that focus on one type of linguistic competence, like PETE, may be one possible path in the direction of more such evaluation schemes. To our knowledge, PETE is the first task to use crowd-sourcing with non-expert, native speaker judgements for parser evaluation, which has traditionally had to rely on trained experts because of the complexity of labeling parse trees.

1.2 Other approaches

There are currently two main approaches in the field of parser evaluation. The treebank based measures introduced nearly two decades ago (Black et al 1991) compare phrase-structure bracketings or dependency links produced by the parser with the ones in an annotated corpus, or "treebank". A second, more recent strand of parser evaluation methods is based on grammatical dependency relations, proposed for ease of use by end users and suitable for parser evaluation. These include the grammatical relations (GR) of Carroll et al (1999), the PARC representation (King et al 2003), and Stanford typed dependencies (SD) (De Marneffe et al 2006) (see Bos et al (2008) for other proposals).

Compared to the first approach, treebank based evaluation methods, parser evaluation using short textual entailments has the following advantages:

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 ($19.50) and each annotation is based on the unanimous agreement of at least three workers.

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, however, 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 2007). However, such conversions are noisy due to fundamental cross-formalism differences as well as inconsistencies in the original treebank (Hockenmaier 2003; Cer et al 2010), compounding the already mentioned problems of treebank based evaluation. Clark and Curran (2007) similarly found an upper bound of 85% accuracy when translating between two grammatical dependency based formalisms for parser 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.

The second approach, evaluation methods based on grammatical dependency relations, uses a set of binary relations between words in a sentence as the primary unit of representation. These methods share some common motivations: usability by people who are not (computational) linguists and suitability for relation extraction applications. Furthermore, they more closely represent semantics than treebank constituency or dependency measures, and various types of parsers are capable of producing dependencies as an interchange format. 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 was inspired by such methods, but 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:

Ease of use: Each of the three proposals of grammatical dependency relations mentioned in this paper (GR, PARC, SD) uses a different set of binary 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 this study we were able to achieve unanimous agreement among 3 or more untrained annotators for close to half of the entailments generated for PETE without any training, correction or adjudication.

Relevance: Though grammatical dependency schemes represent more semantic information than treebank annotation, they still give equal weight to dependencies of differing semantic importance, for example determiners compared to verbal arguments. They are typically used in the aggregate, so that evaluation is weighted towards more frequent dependency types, not necessarily more important ones.[1] When labeled dependencies are used, differences in annotation style can affect evaluation, for example whether

[1] The collapsed and propagated version of Stanford dependencies somewhat mitigates this problem and this is the parser output representation we chose to use as input to the example entailment module of Section 7.

a dependency is labeled *amod* or *advmod*. In contrast, PETE is designed to focus on semantically relevant types, and is label-free.

1.3 Challenges

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. In Section 7 we present a standard entailment module which can be used with any parser that can output Stanford typed dependencies.

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 make mistakes. We hope this methodology will uncover weaknesses that the next generation of parsers can focus on.

1.4 Summary

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. Section 3 presents the participating systems, their methods and results. Section 4 presents the best scoring Cambridge system in more detail. Sections 5 and 6 give a detailed error analysis of the C&C parser and the entailment system used in the Cambridge system. Section 7 introduces a standard entailment system for Stanford typed dependencies and evaluates some example systems for several state of the art parsers. Section 8 summarizes our contribution.

2 Dataset

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

– Identify syntactic dependencies challenging to state of the art parsers.
– Construct short entailment sentences that paraphrase those dependencies.
– 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". For example, the relation between *wrote* and *paper* in *the paper we wrote* is an example of extraction from a reduced relative clause. 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[2].
– 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 gold parse. (The gold parses were available since all sentences came from existing treebanks.) 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 affect downstream applications. These were chosen to turn into entailments in the next step.

2.2 Constructing Entailments

Entailment construction was performed manually by annotators trained to interpret phrase structure and dependency parser outputs. Each hypothesis sentence was based on the relationship between two content words that have a syntactic dependency. The content word pairs were chosen to demonstrate differences between a parser output and the gold parse. All true and false hypotheses generated in this fashion that passed the annotator agreement test

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

as described in Section 2.3, were added to the dataset. The instructions for the annotators were:

1. Idenfity a sentence where at least one parser gives a different parse tree than the gold parse. Use this sentence as the text of a text–hypothesis pair.
2. Identify content words (defined as nouns, verbs, adjectives, and adverbs) in the sentence which have different syntactic heads or different relations to their syntactic heads in the parser output and the gold parse. If the syntactic head is a function word then consider the closest content word ancestor.
3. For each word–head pair identified in the previous step, construct a minimal hypothesis sentence that expresses the same syntactic relation between the two as was observed in the source parse tree. If the pair comes from the gold parse this generates a TRUE entailment, otherwise this generates a FALSE entailment.
4. If the two content words are not sufficient to construct a grammatical sentence use one of the following techniques:
   – Complete the mandatory elements using the words "somebody" or "something" (e.g. to express the subject-verb dependency in "John kissed Mary." construct the hypothesis "John kissed somebody.").
   – Make a passive sentence to avoid using a spurious subject (e.g. to express the verb-object dependency in "John kissed Mary." construct the hypothesis "Mary was kissed.").
   – Make a copular sentence or use existential "there" to express noun modification (e.g. to express the noun-modifier dependency in "The big red boat sank." construct the hypothesis "The boat was big." or "There was a big boat.").

As an example consider the sentence "John slept in the bed." Let us consider what entailments can be constructed from this sentence, assuming we have the gold parse tree. The three content words are "John", "slept", and "bed". "Slept" is the root of the sentence and has no head. The head of "John" is "slept", so we generate the hypothesis (John slept). The syntactic head of "bed" is "in", a function word, so we include the content word ancestor "slept", resulting in the hypothesis (Somebody slept in the bed). Note that we introduce "Somebody", as in step 4 of the instructions, to make the hypothesis a complete sentence without including more constituents from the text. These are the only two hypothesis sentences that can be generated for this sentence.

In general the number of content words gives an upper bound on the number of entailments (text–hypothesis pairs) generated from a sentence. However not every entailment generated in this way made it into the final dataset because we only included entailments related to parser errors, as described in the previous section, and we filtered ones that did not result in unanimous annotator agreement as described in the next section.

The emphasis on having two content words per entailment reflects the relationship between PETE and grammatical dependency schemes. Entailments were constructed manually, although in the future we hope to develop automatic methods. In the first edition of PETE we did not measure inter-annotator agreement on entailment generation; based on the guidelines there should in general be a single well-defined H for each pair of content words, although issues such as embedded clauses and noun modification may need to be more carefully analyzed from an entailment generation perspective in the future to be sure of this.

A list of the content word pairs used to construct the entailments was provided to participants as background information, but the list was not accessible to the entailment systems developed by the participants. Thus entailment decisions could not be facilitated by limiting interrogation of parser output to specific lexical items (although some systems did independently choose to prioritize general *categories* of words or relations in their decisions).

2.3 Filtering Entailments

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

- Each entailment was tagged by 5 untrained annotators from the Amazon Mechanical Turk crowd-sourcing service.
- The results from the annotators whose agreement with the "silver" standard truth values fell below 70% were eliminated.
- The entailments for which there was unanimous agreement of at least 3 annotators were kept.

The second step was necessary to eliminate annotators that were answering questions randomly. The "silver" standard truth values were "yes" for entailments generated from parses agreeing with the gold parses, and "no" for entailments generated from incorrect parses (recall that the gold parses were available from existing treebanks). We call these truth values silver rather than gold since they became part of the gold standard only when unanimously agreed to by three annotators. Though not perfect, the 70% measure provided a simple benchmark to detect annotators answering randomly.

The annotators were allowed to give "Not sure" answers which were later grouped with the "No" answers during evaluation. 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.

2.4 Dataset statistics

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

Approximately half of the final entailments were based on sentences from the Unbounded Dependency Corpus, a third were from the Brown section of the Penn Treebank, and the remainder were from the Charniak sentences. Table 2 gives the breakdown of the original list of entailments and the ones retained after the annotation filtering, according to the text source and the entailment value. Between 1/3 and 1/2 of the original entailments were kept in the final dataset in each category.[3]

	Pre-filter			**Post-filter**		
	All	Y	N	All	Y	N
Unbounded	529	327	202	196	108	88
Brown	335	211	124	124	61	63
Charniak	116	65	51	47	22	25
Total	980	603	377	367	191	176

Table 2 Breakdown of data by source and entailment value before and after the annotation filter.

Table 3 lists the most frequent grammatical relations and constructions encountered in the entailments before and after the annotation filter. Note that the resolution of each entailment may rely on multiple grammatical phenomena, thus the numbers add up to more than 100%.

GR	**Pre-filter**	**Post-filter**
Direct object	48%	42%
Nominal subject	44%	33%
Reduced relative clause	25%	21%
Relative clause	20%	14%
Passive nominal subject	17%	7%
Open clausal complement	6%	2%
Clausal complement	6%	2%
Prepositional modifier	6%	5%
Adverbial modifier	2%	3%
Object of preposition	2%	5%

Table 3 Most frequent grammatical relations and constructions encountered in the entailments before and after the annotation filtering process.

The two groups of entailments that most often failed the inter-annotator agreement filter involved clausal complements and passivization. Constructing

[3] Note that some of the difficult constructions, plus noise in the laypeople's responses meant a large percentage of potential entailments didn't pass the filter, but nevertheless at a nominal cost we were able to create a dataset where all the entailments were unanimously agreed by 3 people, which is not the case for most other commonly used treebanks.

entailments for clausal complements based on two content words as described in Section 2.2 was sometimes challenging for the entailment generators and confusing to the annotators. Annotators failed to reach a unanimous agreement on examples like the following:

Text: He found a jar of preserved tomatoes and one of eggs that they had meant to save.
Hypothesis: Somebody had meant to save one.

Passivization, which was used when constructing entailments from a verb object pair also proved to be confusing at times. Annotators also failed to reach unanimous agreement on some examples like the following:

Text: But he kept Fruit of the Loom Inc., the underwear maker that he still controls and serves as chairman and chief executive.
Hypothesis: The maker is served.

3 Task Results

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 with substates	0.5017	0.5395	0.2628	0.3534
390-434-Berkeley 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 with Viterbi Decoding	0.4784	0.4964	0.4359	0.4642

Table 4 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.

Twenty systems from 7 teams participated in the PETE task. Table 4 gives the percentage of correct answers for each system. Twelve systems 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 the Combinatory Categorical Grammar (CCG; Steedman (2000)) formalism. Others used dependency structures produced by MaltParser (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.

The top two scoring systems, Cambridge and SCHWA (University of Sydney), were based on the C&C parser and used a similar approach (though Cambridge used GR output in SD format while SCHWA used the native CCG dependency output of the parser). They achieved almost identical task accuracies, but SCHWA was more accurate on "yes" entailments, while Cambridge was more accurate on "no" entailments, resulting in a higher overall accuracy for Cambridge, but a higher F-score on positive entailments for SCHWA (Table 4). We attribute this difference to the decision criteria used in the entailment systems, which will be discussed in Section 4, but notably the difference suggests that a dependency-based entailment system can be tuned to favour precision or recall.

The following sections describe the Cambridge system in more detail and present detailed error analyses for the parser and the entailment system.

4 The Cambridge System

The best–scoring Cambridge system used the C&C parser (Clark and Curran 2007), which can produce GR output in SD format (see Section 1) using custom tools available with the parser.[4]

The entailment system was very simple, and based on the assumption that H is a simplified version of T, which is true for this task though not necessarily for RTE in general. Let grs(S) be the GRs produced by the parser for a sentence S. The basic intuition is that if grs(H) $\subseteq$ grs(T), then in principle H should be considered an entailment of T. In practice, certain refinements of this basic intuition were required to account for non-matching GRs resulting from grammatical transformations used in entailment construction, or noise in the parse which could be safely ignored.

Three situations were identified in which GRs in H would not exactly match those in T. First, syntactic transformations used in entailment construction could change head-dependent relations. By far the most frequently used transformation in the PETE dataset was passivization.

[4] http://svn.ask.it.usyd.edu.au/trac/candc

Second, the introduction of dummy words and transformations during entailment construction meant that H could contain tokens not present in T. This included pronouns such as “somebody” and “something”, auxiliary verbs introduced by passivization, and expletive subjects. In addition, determiners were sometimes introduced or changed, e.g. “prices” to “the prices”.

Third, the parses of T and H might be inconsistent in a way incidental to the target entailment. Consider the sentence pair T: *I reached into that funny little pocket that is high up on my dress.* $\Rightarrow$ H: *The pocket is high up on something.* The intended focus of the evaluation is the relation between “pocket” and “high”. As long as the parser analyzes “pocket” as the subject of “high”, we want to avoid penalizing it for, say, attaching the PP beginning with “up on” differently in T and H.

To address these issues the system used a small set of heuristics. First, it ignored any GR in grs(H) containing a token not in T. This addressed the pronouns, passive auxiliaries, expletive subjects, and determiners. Second, it equated passive subjects with direct objects. Similar heuristics could be defined to accommodate other transformations, but only this one was implemented, based on examination of the development set.

Third, when checking whether grs(H) $\subseteq$ grs(T), only the core relations subject and object were considered. The intention was that incidental differences between the parses of T and H would not be counted as errors. These GR types were chosen based on examination of the entailments in the development set, but the system could easily be reconfigured to focus on other relation types, e.g. PP relations for a PP-attachment task.

Finally, the system required grs(H) $\cap$ grs(T) to be non-empty (no vacuous positives), but did not restrict this criterion to subjects and objects.

The system used a PTB tokenizer[5] for consistency with the parser's training data. The morpha lemmatizer (Minnen et al 2000), which is built into the C&C tools, was used to match tokens across T and H, and all tokens were converted to lowercase. If the parser failed to find a spanning analysis for either T or H, the entailment decision was “no”. The full pipeline is shown in Figure 1.

Comparing the Cambridge system with the SCHWA system, Cambridge was more accurate on “no” entailments and SCHWA on “yes” entailments. We believe this is because both systems required at least one matching relation between T and H for a “yes” answer, but Cambridge additionally answered “no” if any core relation (subject or object) was present for H but not for T. Thus Cambridge permitted fewer false positives than SCHWA.

5 Error Analysis

Table 4 includes the results for the Cambridge system on the test set. On the development set the system achieved an overall accuracy of 66.7%. On positive entailments, precision was 0.7857, recall was 0.5789, and F-score was 0.6666.

[5] http://www.cis.upenn.edu/˜treebank/tokenizer.sed

Tokenize T and H with PTB tokenizer

⇓

Parse T and H with C&C parser

⇓

Lowercase and lemmatize all tokens

⇓

Discard any GR in grs(H) containing a token not in T

⇓

Convert passive subjects to direct objects

⇓

"yes" if core(H) $\subseteq$ core(T) and grs(H) $\cap$ grs(T) $\neq \emptyset$,
"no" otherwise

Fig. 1 Full pipeline for C&C parser and entailment system. core(S): the set of core (subject and object) GRs in grs(S).

Table 5 lists the frequency of various grammatical relations in the development and test set instances where the Cambridge system made mistakes. A comparison with Table 3 shows direct objects and reduced relative clauses to be frequent causes of error.

GR	**Dev Set**	**Test Set**
Reduced relative clause	45%	36%
Direct object	18%	51%
Relative clause	18%	—
Nominal subject	9%	20%
Passive nominal subject	9%	7%
Adverbial modifier	9%	—
Prepositional modifier	9%	—
Conjunct	9%	—
Object of preposition	—	7%

Table 5 Frequency of grammatical relations in entailment instances that got wrong answers from the Cambridge system.

Table 6 further breaks down the results on the development set to show how different types of parser and entailment system errors contributed to incorrect answers. In the majority of cases the parser and entailment system worked together to find the correct answer as expected. For example, for T: *Trading in AMR shares was suspended shortly after 3 p.m. EDT Friday and didn't resume.* ⇒ H: *Trading didn't resume.*, the parser produced three GRs for H (tokens are shown lemmatized and lowercase): `(nsubj resume trading)`, `(neg do n't)`, and `(aux resume do)`. All of these were also in grs(T), and the correct "yes" decision was made. For T: *Moreland sat brooding for a full minute, during which I made each of us a new drink.* ⇒ H: *Minute is made.*, the parser produced two GRs for H. One, `(auxpass make be)`, was ignored because the passive auxiliary "be" is not in T. The second, passive subject GR `(nsubjpass make minute)` was equated with a direct object `(dobj make minute)`. This GR was not in grs(T), so the correct "no" decision was made.

In some cases a correct "yes" answer was reached via arguably insufficient positive evidence. For T: *He would wake up in the middle of the night and fret about it.* ⇒ H: *He would wake up.*, the parser produces incorrect analyses for the VP "would wake up" for both T and H. However, these GRs are ignored since they are non-core (not subject or object), and a "yes" decision is based on the single GR match `(nsubj would he)`. This is not entirely a lucky guess, since the entailment system has correctly ignored the faulty analyses of "would wake up" and focused on the role of "he" as the subject of the sentence. However, especially since the target was the relation between the subject "he" and the lexical verb "wake", more positive evidence would be desirable. Of the 22 correct "yes" decisions, only two were truly lucky guesses in that the single match was a determiner; all others had at least one core match.

Type	**FN**	**FP**	**Total**
Unbounded dependency	9	1	10
Other parser error	6	2	8
Entailment system	1	3	4
Total	16	6	22

Table 6 Error breakdown on the development set. FN: false negative, FP: false positive.

Table 6 shows the breakdown of errors. The largest category was false negatives due to unbounded dependencies not recovered by the parser, for example T: *It required an energy he no longer possessed to be satirical about his father.* ⇒ H: *Somebody no longer possessed the energy.* Here the parser fails to recover the direct object relation between "possess" and "energy" in T. It is known that parsers have difficulty with unbounded dependencies (Rimell et al 2009), so this result is not surprising. Among the unbounded dependencies, one gold standard entailment was particularly difficult because it involved two layers of extraction; this was T: *Index-arbitrage trading is "something we want to watch closely," an official at London's Stock Exchange said.* ⇒ H: *We want to watch index-arbitrage trading.* Here, recovering the entailment requires not only correctly parsing the relative clause headed by "something", but also resolving the reference of "something" as "trading", a compound task that few modern syntactic parsers attempt. Nevertheless, this is a legitimate entailment, constructed according to the guidelines in Section 2.2.

The next category was other parser errors. This is a miscellaneous category including e.g. errors on coordination, parenthetical elements, identifying the head of a clausal subject, and one error due to the POS tagger. For example, for T: *Then at least he would have a place to hang his tools and something to work on.* ⇒ H: *He would have something to work on.*, the parser incorrectly coordinated "tools" and "something" for T, making "something" appear to be an object of "hang". As a result `(dobj have something)` was in grs(H) but not grs(T), yielding an incorrect "no".

Four errors were due to the entailment system rather than the parser; these will be dicsussed in Section 6.

6 Entailment System Evaluation

The utility of PETE as a parser evaluation tool depends on the availability of appropriate entailment systems, since the entailment system acts as an intermediary between the parser and the PETE dataset. We believe the appropriate role of an entailment system is to be fully transparent with respect to the parser output, that is, to faithfully reflect whether the parser has made attachment decision(s) for T which entail a "yes" answer for H, neither introducing nor correcting any errors. Recalling the example T: *The man with the hat was tired*, if the parser analyzes "man" as the subject of "was tired", the entailment system should pass along a "yes" answer to H-1: *The man was tired*, and a "no" answer to H-2: *The hat was tired.* If the entailment system is transparent in this way, then a correct answer on H indicates a correct parse (with regard to the content of H) and an incorrect answer on H indicates a parser error.

It is easy to imagine a variety of non-transparent entailment systems. For example, an entailment system that always answers "yes" regardless of parser output will yield an accuracy score exactly equal to the "always yes" baseline, no matter how well or poorly the underlying parser has performed. On the other hand, an entailment system that is allowed to override parser output based on background knowledge or reasoning heuristics would also be non-transparent because it could improve on the parser's decisions. Note we do not say there is no role in NLP for such an entailment system, only that it would not transparently convey parser accuracy on the PETE dataset. Moreover, there may be room for an intermediate, semi-transparent variety of entailment system that cannot override attachments made by the parser, but can add information.

We can say the PETE task is a valid parser evaluation tool if it is possible to construct an appropriate entailment system for any given parser. We do not attempt to evaluate whether the entailment systems of all participating systems were appropriate, but as a case study, we consider whether the Cambridge entailment system was an appropriate tool for evaluating the C&C parser on the PETE dataset. A more generalized entailment system is described in Section 7.

We use two oracle experiments to isolate and evaluate the performance of the Cambridge entailment system. The first oracle experiment uses gold-standard GRs rather than automatic parser output as input to the entailment system. Assuming the Cambridge GR-based approach is valid, then given gold-standard GRs for T and H, we expect an appropriate entailment system to result in 100% accuracy on the task evaluation (because all parses are correct, and the entailment system should faithfully pass along the correct analyses). To perform this experiment we manually annotated all T and H sentences in the development set with gold-standard GRs. Using the Cambridge entailment system with the gold GRs resulted in a task accuracy score of 90.9%, which can therefore also be considered the entailment system accuracy score in this experiment.

Of the six errors made by the system, three (two FN and one FP) were due to transformations between T and H which changed the GR label or head. For example, consider T: *Occasionally, the children find steamed, whole-wheat grains for cereal which they call "buckshot".* ⇒ H: *Grains are steamed.* In T, "steamed" is a prenominal adjective with "grains" as its head; while in H, it is a passive with "grains" as its subject. The entailment system did not account for this transformation. In principle it could account for any transformation which can be expressed as a rule on GRs, in the same way it accounts for passivization. In practice, the only way to guarantee that an entailment system can account for all transformations between T and H in the dataset is for the complete list of possible transformations to be documented in the entailment generation guidelines and made available to developers.

Two errors (both FP) occurred when GRs involving a non-core relation or a pronoun introduced in H, both of which the system ignored, were crucial for the correct entailment decision. Since the development decision to ignore non-core relations and pronouns introduced in H was made for overall accuracy, errors are inevitable in these circumstances, but they are a small percentage of sentences in the dataset.

The final error was on the difficult unbounded dependency discussed in Section 5, T: *Index-arbitrage trading is "something we want to watch closely," an official at London's Stock Exchange said.* ⇒ H: *We want to watch index-arbitrage trading.* The gold GRs represent the unbounded dependency correctly, but nevertheless do not provide the necessary information to resolve the reference of "something" as "trading".

The second experiment compared the Cambridge entailment system with an oracle entailment system, i.e. manual judgements on whether T entails H given the parser's analysis. We used the GRs automatically generated by C&C for the development set, and manually decided for each sentence whether T entailed H based *only* on the automatic parser analysis. We then compared this manual analysis with the automatic entailment system decisions, to determine how transparent the entailment system was.

Based on the manual analysis, we found the Cambridge entailment system made six errors on the development set, when using automatically generated C&C output. Two errors were in the parser's favor, i.e. the parser analysis was incorrect, but the entailment system "corrected" the error; and four were to its detriment, i.e. the parser analysis was correct but the entailment decision was incorrect. Accuracy of the entailment system was 90.9% on this measure, consistent with the results of the previous oracle experiment (though the erroneous sentences were not identical).

Among the two errors in the parser's favor, one involved T: *They wanted to see what his back felt like – the General's.* ⇒ H: *Somebody wanted to see what his back felt like.* The parser analyzed the phrase "what his back felt like" incorrectly, but made the same error for both T and H, so that the GRs matched. Only by manual analysis could the error be found. The other error was on a sentence where the attachment decision was incorrect, but there was a single GR match on a determiner.

The four entailment system errors to the detriment of the parser correspond to the "Entailment system" row in Table 6. Three of these errors also occurred in the first oracle experiment and have been discussed above. The fourth resulted from a POS change between T and H for T: *There was the revolution in Tibet which we pretended did not exist.* ⇒ H: *The pretended did not exist.* The crucial GR for finding a "no" answer was `(nsubj exist pretended)` in grs(H), but the entailment system ignored it because the lemmatizer did not give "pretend" as the lemma for "pretended" as a noun. This type of error might be prevented by answering "no" if the POS of any word changes between T and H, although the implementation would be non-trivial since word indices may also change.[6]

Note that the 90.9% accuracy figure for the entailment system based on the manual analysis in the second oracle experiment does not reflect *parser* accuracy: when the parser made an error on a crucial dependency, leading to an incorrect entailment decision, we judged the entailment system to be correct, since it faithfully passed along the parser's error for the purpose of parser evaluation. If the C&C parser had been coupled with an oracle (i.e. fully manual) entailment system, it would have achieved 69.7% accuracy on the development set, compared to the 66.7% it achieved with the automatic entailment system.

The high accuracy levels for the simple Cambridge entailment system, which was over 90% accurate at passing along the performance of C&C on the PETE dataset, are very promising. Some additional accuracy could be recovered in the future if parser and entailment system developers have access to improved documentation of the linguistic transformations permitted in entailment generation. This is a positive result for the validity of the PETE task.

7 A Generalized Entailment System

In order to facilitate further research on this task, we implemented a more modular and generalized entailment system based on the Cambridge system. By applying similar heuristics and search methods, we are able to replicate the top score on the task, and are also able to compare different parsing paradigms using a single entailment system, making for a more level playing field.

This system takes Stanford dependencies as input and thus integrates with several publicly available parsers which are capable of producing Stanford typed dependencies. The parsers used were the Berkeley Parser (Petrov and Klein 2007), Charniak Parser (Charniak and Johnson 2005), Collins Parser (Collins 2003), C&C Parser (Clark and Curran 2007), Malt Parser (Nivre et al

[6] There were eight POS changes in the development set, most of which did not result in errors on evaluation. Note also that this particular H is ungrammatical English. Recall that the negative H sentences were derived from genuine parser errors; it was not always possible to construct grammatical sentences corresponding to such errors, though we will consider constraining all H sentences to be grammatical in future work.

Parse T and H

$\Downarrow$

Convert to Stanford collapsed dependencies with propagation of conjunct dependencies

$\Downarrow$

Lowercase and lemmatize all tokens

$\Downarrow$

Convert passive subjects to direct objects

$\Downarrow$

Extract core dependencies: `nsubj`, `dobj`, `prep-`$*$

$\Downarrow$

"yes" if core(H) $\subseteq$ core(T) and core(H) $\cap$ core(T) $\neq \emptyset$,
"no" otherwise;
allow matching of dummy pronouns in H with lexical items in T

Fig. 2 Full pipeline for Stanford dependencies entailment system. core(S): the set of core (subject, object and preposition) GRs in grs(S).

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. MaltParser and MSTParser were trained on the Stanford dependencies format of Penn Treebank as described in (Cer et al 2010), and C&C was trained on CCGbank (Hockenmaier 2003).

Stanford typed dependencies come in several varieties (De Marneffe and Manning 2008). The Cambridge system used the option of tree-breaking dependency types *ref*, *xsubj*, and *pobj* and propagation of conjunct dependencies, but no collapsed dependencies. The generalized entailment system used Stanford dependencies with fully collapsed dependencies (including tree-breaking dependencies) and propagation of conjunct dependencies. Using the fully collapsed dependencies was intended to allow improved matching of various relations, especially prepositions, between T and H.

The outputs of all parsers except C&C were converted to the Stanford collapsed dependency representation with propagation of conjunct dependencies using the Stanford Parser. Because the C&C parser does not produce a representation suitable for conversion with the Stanford tools, we converted the C&C output to Stanford dependencies using custom tools, as in Section 4.

To decide on entailments both the test and hypothesis sentences were parsed. All the words in T and H were lemmatized and lowercased after parsing. We apply the same heuristics as in the Cambridge system for active-passive conversion and dummy word matching. We then consider the core dependency types `nsubj`, `dobj`, and `prep-`$*$ when comparing T and H (the use of the collapsed dependency representation making it possible to consider prepositions as a core relation). If there is at least one core dependency in H, and all core dependencies in H are also found in T, the decision is "yes". If the core H dependencies are not found in T the decision is "no". The full pipeline is shown in Figure 2.

Table 7 lists the results achieved. There are significant differences in the entailment accuracies of systems that have comparable unlabeled attachment scores (UAS), with UAS derived from parser output in the CoNLL representation. 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. The difference between UAS and PETE scores reflects the indifference of treebank based measures like UAS to the semantic significance of various dependencies and their impact on potential applications.

Note that different conversion steps en route to Stanford dependencies may mean that results are not exactly comparable for all parsers. Nevertheless, the PETE score provides a notably different perspective on the parser results. We note that the two dependency parsers, MaltParser and MSTParser, show lower accuracies than the constituent parsers; this is consistent with the results in (Cer et al 2010). No UAS is available for C&C, which does not produce CoNLL style output.

System	**PETE**	**UAS**
C&C Parser	73.42%	–
Collins Parser	71.43%	91.6
Berkeley Parser	71.10%	91.2
Charniak Parser	68.44%	93.2
Stanford Parser	67.11%	90.2
MaltParser	64.12%	89.8
MSTParser	62.46%	92.0

p-value	**Coll**	**Berk**	**Char**	**Stan**	**Malt**	**MST**
C&C Parser	.5663	.4567	.0966	**.0351**	**.0032**	**.0004**
Collins Parser		1.0	.2893	.1056	**.0108**	**.0012**
Berkeley Parser			.2299	.0667	**.0192**	**.0024**
Charniak Parser				.6582	.1485	**.0421**
Stanford Parser					.3134	.1149
MaltParser						.5595

Table 7 Example systems: The first table gives the performance on the PETE test set, and the unlabeled attachment score on section 23 of the Penn Treebank. The second table gives the p-values for the differences between PETE scores based on the McNemar test (Dietterich 1998). Statistically significant differences ($p < .05$) are indicated with bold typeface.

8 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 make mistakes 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 the natural competence of annotators in comprehending sentences rather than their imperfect performance on artificial tagging tasks.

Multiple systems, including the examples described in Section 7 achieved good results on the PETE task using state-of-the-art parsers and simple entailment systems. The analysis of the Cambridge entailment system showed it to have accuracy of approximately 90% as a tool for evaluating the C&C parser, or potentially any parser producing GRs, on the PETE development data. This result is perhaps even more important than the task scores since it suggests that PETE is worth pursuing as a parser evaluation approach.

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 Stephan Oepen and Anna Mac for their careful analysis and valuable suggestions. Önder Eker and Zehra Turgut contributed to the development of the PETE task. Stephen Clark collaborated on the development of the Cambridge system. We would also like to thank Matthew Honnibal for discussion of the SCHWA system and contribution to the entailment system analysis.

References

Black E, Abney S, Flickenger D, Gdaniec C, Grishman R, Harrison P, Hindle D, Ingria R, Jelinek F, Klavans J, 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, Morgan Kaufmann Pub, p 306

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

Bos J, et al (eds) (2008) Proceedings of the Workshop on Cross-Framework and Cross-Domain Parser Evaluation, URL `http://lingo.stanford.edu/events/08/pe/`, in connection with the 22nd International Conference on Computational Linguistics

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

Cer D, de Marneffe MC, Jurafsky D, Manning CD (2010) Parsing to stanford dependencies: Trade-offs between speed and accuracy. In: 7th International Conference on Language Resources and Evaluation (LREC 2010), URL `http://nlp.stanford.edu/pubs/lrecstanforddeps\_final\_final.pdf`

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

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

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

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

De Marneffe M, Manning C (2008) Stanford typed dependencies manual. URL http://nlp.stanford.edu/software/dependencies-manual.pdf

De Marneffe M, MacCartney B, Manning C (2006) Generating typed dependency parses from phrase structure parses. In: LREC 2006

Dickinson M, Meurers WD (2003) Detecting inconsistencies in treebanks. In: Proceedings of the Second Workshop on Treebanks and Linguistic Theories (TLT 2003), Växjö, Sweden, pp 45–56, URL \url{http://ling.osu.edu/~dickinso/papers/dickinson-meurers-tlt03.html}

Dietterich T (1998) Approximate statistical tests for comparing supervised classification learning algorithms. Neural computation 10(7):1895–1923

Erk K, McCarthy D, Gaylord N (2009) Investigations on word senses and word usages. In: Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 1-Volume 1, Association for Computational Linguistics, pp 10–18

Hockenmaier J (2003) Data and models for statistical parsing with combinatory categorial grammar. PhD thesis, University of Edinburgh

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

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

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

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

McCarthy D, Navigli R (2007) Semeval-2007 task 10: English lexical substitution task. In: Proceedings of the Fourth International Workshop on Semantic Evaluations (SemEval-2007), Association for Computational Linguistics, Prague, Czech Republic, pp 48–53, URL http://www.aclweb.org/anthology/W/W07/W07-2009

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

Minnen G, Carroll J, Pearce D (2000) Robust, applied morphological generation. In: Proceedings of INLG, Mitzpe Ramon, Israel

Nivre J, Hall J, Kübler S, McDonald R, Nilsson J, Riedel S, Yuret D (2007a) The CoNLL 2007 shared task on dependency parsing. In: Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL, vol 7, pp 915–932

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

Petrov S, Klein D (2007) Improved inference for unlexicalized parsing. In: Proceedings of NAACL HLT 2007, pp 404–411

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

Snyder B, Palmer M (2004) The English all-words task. In: ACL 2004 Senseval-3 Workshop, Barcelona, Spain, URL http://www.cse.unt.edu/~rada/senseval/senseval3/proceedings/pdf/snyder.pdf

Steedman M (2000) The Syntactic Process. The MIT Press, Cambridge, MA