TAG Parser Evaluation using Textual Entailments

Pauli Xu[1], **Robert Frank**[1], **Jungo Kasai**[1], and **Owen Rambow**[2]

[1]Yale University ({pauli.xu, bob.frank, jungo.kasai}@yale.edu)
[2]Columbia University (rambow@ccls.columbia.edu)

Abstract

Parser Evaluation using Textual Entailments (PETE, Yuret et al. (2013)) is a restricted textual entailment task designed to evaluate in a uniform manner parsers that produce different representations of syntactic structure. In PETE, entailments can be resolved using syntactic relations alone, and do not implicate lexical semantics or world knowledge. We evaluate TAG parsers on the PETE task, and compare our results to the state-of-the-art. Our TAG parser combined with structural transformations to compute entailments outperforms the CCG-based results on the development set, though it falls behind these results on the test set. The CCG parser makes use of a number of heuristics for entailment comparison, however. Adding such heuristics to our best TAG parser yields state-of-the-art results on the test set when using accuracy as a metric. This sensitivity to heuristics suggests that the PETE task may suffer from an unrepresentative development set, and that we need to improve upon formalism-independent parsing evaluation methods.

1 Introduction

There has been a flurry of recent work, involving neural network architectures, on parsing that has improved performance across a variety of frameworks that make different assumptions about the target output for the parsing process: Dependency grammar (Chen and Manning, 2014; Dyer et al., 2015; Andor et al., 2016; Kuncoro et al., 2017; Dozat and Manning, 2017), Combinatory Categorial Grammar (CCG) (Xu et al., 2015; Ambati et al., 2016; Lewis et al., 2016), Tree Adjoining Grammar (TAG) (Kasai et al., 2017), Constituent structure (Dyer et al., 2017; Kuncoro et al., 2017)). However, it is as yet unknown the degree to which these improvements in parsing scores contribute to downstream NLP tasks. Moreover, since the different frameworks make different representational assumptions about the target of the parsing process, these results are not directly comparable.

Parser Evaluation using Textual Entailments (PETE) is a shared task from the SemEval-2010 Exercises on Semantic Evaluation (Yuret et al., 2013). The task was intended to evaluate syntactic parsers across different formalisms, focusing on entailments that could be determined entirely on the basis of the syntactic representations of the sentences that are involved, without recourse to lexical semantics logical reasoning or world knowledge. For instance, syntactic knowledge alone tells us that the sentence *Peter, who loves Mary, left the room* entails *Peter left the room* and *Peter loves Mary* but not, for example, that *Peter knows Mary* or that *Peter was no longer in the room.*

In this paper, we apply a number of TAG parsers to the PETE task. In the next section, we discuss the PETE task in further detail. Then in Section 3 we describe how we apply TAG parses to this task. Doing so requires a means of determining whether one TAG derivation entails another syntactically. We do this through a set of task-independent, linguistically-motivated transformations. After reviewing the TAG supertaggers and parsers we evaluate in Sections 4 and 5, we discuss our results in Section 6. We demonstrate that improvements in TAG parsing and supertagging do indeed contribute to improvements in the extrinsic PETE task, reaching state-of-the-art results in accuracy and near state-of-the-art in f-measure. In particular, we compare our results to the top-scoring systems of SemEval-2010, Cambridge (Rimell and Clark, 2010) and SCHWA (Ng et al., 2010), both based on the Clark and Curran (2007) CCG parser, as well as a later system based on an HPSG-Minimal Recursion Semantics parser (Lien, 2014). We also conduct an error analysis and discuss limitations of TAG parsing in the context of this task.

2 The PETE Task

PETE (Yuret et al., 2013) is a restricted instance of the recognizing textual entailment (RTE) task, aimed at evaluating syntactic parsers. As in other RTE tasks, the task includes a set of *Text-Hypothesis* pairs, for which a system must determine whether or not the content of the Text entails the content of the Hypothesis. For PETE, the texts are individual sentences that were drawn from one of three sources: the Unbounded Dependency Corpus (Rimell et al., 2009), the Brown section of the Penn Treebank, and a list of sentences in the Penn Treebank on which the Charniak parser (Charniak and Johnson, 2005) performed poorly. A sentence S from these sources were selected as a candidate Text if S was misparsed by at least one phrase structure or dependency parsers that was state-of-the-art in 2009.

Given a candidate Text T, an associated Hypothesis H was constructed by identifying a pair of content words in T whose syntactic relationship is implicated in the difference between the gold parse and the incorrect parse. These words were then used to form a minimal sentence. If such a sentence involved a verb that required additional arguments, these could be filled in with indefinite expressions (*somebody*, *someone*, and *something*) or the verb could be passivized. Similarly, in the case of a head and modifier, a copular sentence, or one involving existential *there*, could be constructed. The resulting T-H pair would then be assigned the label 'YES' if the content words in H stand in a relation that is also present in the gold parse of T, and is otherwise assigned the label 'NO'. Each of the resulting T-H pairs were then given to five untrained annotators on Amazon Mechanical Turk and was retained in the dataset if three of them agreed on the presence or absence of the entailment. This left a dataset containing 367 T-H pairs (of which 51.83% were labeled 'YES'). These were then randomly divided into a development set containing 66 sentences and a test set containing 301 sentences. For more details on the construction process, see Yuret et al. (2013).

3 Applying TAG Parsing to PETE

Our TAG-based PETE system determines the entailment status of a T-H pair through the following four steps:

1. T and H are tokenized using the NLTK tokenizer (Bird et al., 2009).
2. T and H are supertagged and parsed, yielding derivation trees D_T and D_H.
3. Structural transformations are applied to D_T to yield a modified derivation graph D'_T.
4. Return 'YES' if D_H is a subderivation of D'_T.

In the following subsection, we describe the properties of the TAG grammar we use for supertagging and parsing and the derivation trees which result from the parser. We then describe the set of structural transformations that are applied to the Text's derivation tree, and define how we determine the subderivation property.

3.1 The TAG Grammar and Derivation Trees

Our experiments make use of the TAG grammar extracted from the Penn Treebank by Chen (2001), and used by the MICA parser (Bangalore et al., 2009). This grammar makes use of the representations developed for TAG starting with the XTAG project (XTAG Research Group, 2001). Positions for a lexical anchor's arguments are labeled with numbers that represent the argument's deep syntactic role. Deep subjects are labeled 0, direct objects and objects of prepositions are labeled 1 and indirect objects are labeled 2. These numbers remain constant across elementary trees that differ with respect to grammatical operations such as passivization and dative shift. In the elementary tree associated with the verb *played* in the passive sentence *The piano was played by Fred*, *the piano* will be labeled as role 1, while *Fred* will be labeled as role 0, just as in the active counterpart *Fred played the piano*.

In the derivation trees that result from a parse with this grammar, the arcs deriving from substitution are labeled by the deep role of substituted argument. Distinct labels in the derivation tree are used for the insertion of co-heads and of adjoining (which are not distinguished by locus of adjoining). Nodes in the derivation tree are associated with a token in the sentence and its corresponding elementary tree. A derivation tree can be thought of as a set of parent-child-relation triples.

Because of the use of deep labels in the grammar, the derivation trees of active and passive sentences will be structurally identical, apart from the identity of the elementary trees. As a result, the most common kind of mismatch between T and H in the PETE task, namely passivization, is handled directly by the parser without any further addition.[1]

3.2 Transformations on the Derivation Tree

As we noted earlier in the description of the PETE task, the sentences comprising T and H may differ in certain syntactically defined ways, giving rise to distinct TAG derivations. As a result, we apply a set of transformations to these derivations to make them more comparable. These were motivated by well-understood properties of TAG derivations as well as by divergences found in the T-H pairs in the development set. The phenomena dealt with by these transformations include NP modification, relative clauses, clausal complementation by predicative auxiliaries, predicative clauses, and coordination.

Modification in a TAG derivation is dealt with via adjoining. In the derivation tree, this will result in the modifier being a child of the head it modifies, with the arc labeled as adjunction. However, there are entailments where we will want to consider a different relationship between these words. For example, in the sentence *I reached into that funny little pocket* (from the PETE development set), the NP modifier *funny* is adjoined to *pocket*. However on a hypothesis like *The pocket is funny*, a predicative sentence, *pocket* is a 0-argument (i.e., subject) of *funny*. To deal with this mismatch, when we find a adjoining dependency between an N-headed tree and an auxiliary tree headed by an adjective, preposition or noun, we add a triple to the derivation tree in the reverse direction with the arc label 0, signifying that the head is a subject argument of the predicate (as it would be in a predicative sentence. This is depicted in the top line of Figure 1.

[1]Note however that the grammar does not retain argument labels across alternations like the causative-inchoative. In *The vase broke*, *the vase* will be argument 0, while in *I broke the vase*, it will be argument 1. Consequently, the entailment from the latter to the former would not follow directly from the parse. We would need PropBank-style argument labeling to recognize such cases.

When the elementary tree that is adjoined to the noun is a relative clause, we do something similar but slightly more complex. In order to determine the role that the reverse dependency should have, we must consult the properties of the relative clause elementary tree. For a subject relative, we add a 0-labeled arc, for an object relative, we add a 1-labeled arc, etc. This is shown on the second line of Figure 1. For adjectival passives that are adjoined to a noun (e.g., *the thrown ball*), we add a 1-labeled arc between the head noun and the verbal head, yielding the inference that *the ball was thrown*.

Sentential complementation in TAG derivations can be analyzed via either adjoining the higher clause into the embedded clause (necessarily so in cases of long-distance extraction from the embedded clause) or substituting the embedded clause in the higher clause. For example, on the third line of Figure 1, we see the derivation tree for *want to watch*, where *want* adjoins into *watch*. In order to normalize this divergence, for adjunction links involving a predicative auxiliary tree, we add a reverse link involving the 1 relation (i.e., that involving sentential complements).

For predicative clauses, like *trading is something we want to watch*, we want to allow for entailments like *we want to watch trading*. Given the transformations we have considered thus far, we will only derive that *we want to watch something* (via the relative clause and predicative auxiliary transformations). However, for auxiliary trees involving nominal predication (*A is B*), we add a derivational link that asserts that A stands in the same relation to predicates which have B as their argument. Thus, since *something* in our example is a 1-argument of *watch*, this rule will assert that *trading* is such an argument as well.

The final structural transformation involves coordination. Under the TAG analysis, VP coordination involves a VP-recursive auxiliary tree headed by the coordinator that includes a VP substitution node (for the second conjunct). We see part of the resulting derivation for the sentence *My host went over and stared out the window* in the top line of Figure 2. In order to allow the first clause's subject argument (as well as modal verbs and negations) to be shared by the second verb, we add the relevant relations to the second verb.[2]

[2]There is some indeterminacy concerning the label of the argument that should be added to the second verb, since this VP tree does not encode what role its subject would be, at

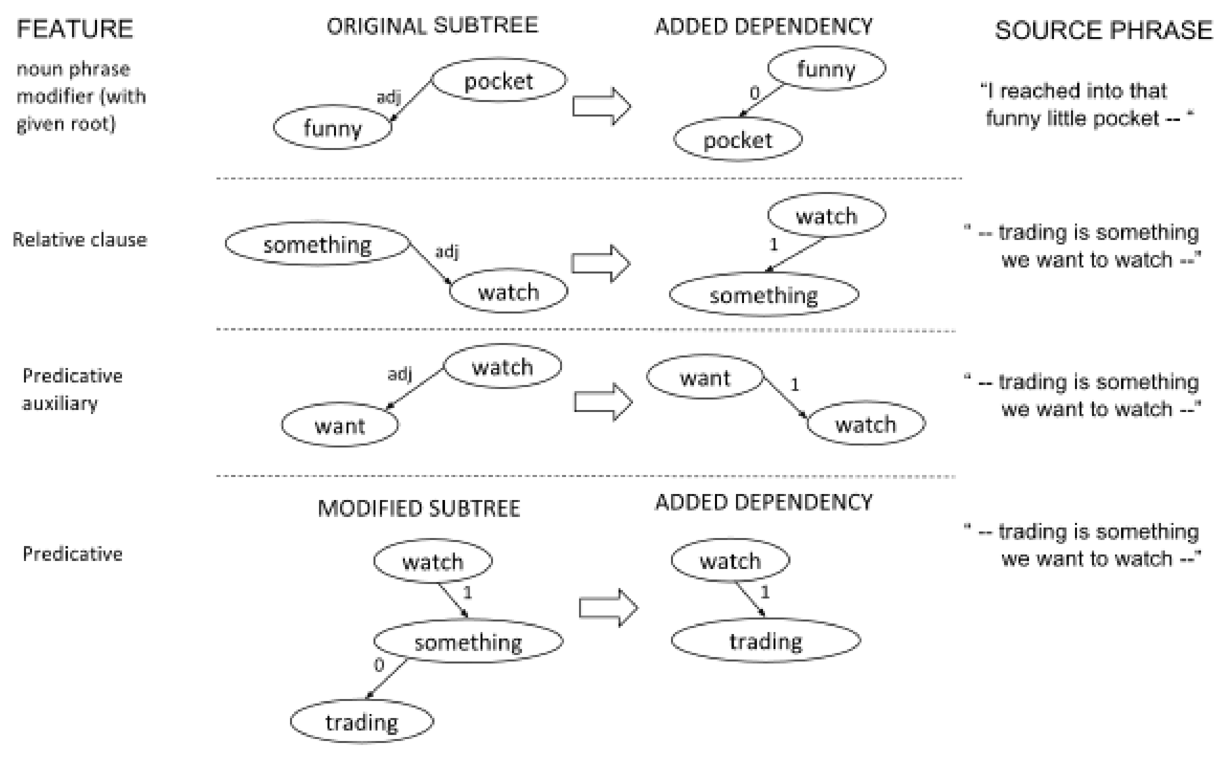

Figure 1: Structural transformations for features NP modifier, relative clause, predicative auxiliary, and predicative clauses

In the case of conjuncts that are the argument of some other predicate, seen in the bottom line of Figure 2 for the sentence *I like John and Mary*, the second conjunct (*Mary* in this example) will inherit any numbered parents of the first conjoined word.

This set of structural transformations is applied in the order in which we have presented it, so that the output of previous transformations can feed subsequent ones.

3.3 Recognizing Entailment

Having applied this series of transformations to the Text's derivation, we determine the presence or absence of an entailment essentially by asking whether the derivation of the Hypothesis is a subset of the derivation of the Text. This cannot however be done in the simplest fashion, because of possible superficial divergences between the derivations, concerning upper and lower case, the location of punctuation and the derivational root, contraction, and the presence of extra functional material (auxiliary verbs or determiners). We therefore ignore these differences when we consider whether D_H is a subderivation of D'_T. We say that D_1 *is a subderivation of* D_2 iff for every $(w_1, w_2, Rel) \in D_1$

1. (Subset) there is a triple $(w'_1, w'_2, Rel) \in D_2$ such that $w_1 \approx w'_1$ and $w_2 \approx w'_2$; or

2. (Ignore root, punctuation and some function words) w_1 or w_2 is ROOT, a punctuation symbol or is lemmatized as one of *be*, *have* or *the* and is adjoined to its parent; or

3. (wildcard indefinites) $w_1 \in \{$*somebody*, *something*, *someone*$\}$ and there is a triple $(w'_1, w_2, Rel) \in D_2$ where w'_1 is a noun.

This definition depends on a near equality relation $\approx$ between holding between words a and b when

1. (lowercase) lowercase(a) = lowercase(b); or

2. (contraction) a =''s' and b ='is' or a ='n't' and b ='not' (or vice versa)

least not in the current grammar. As a result, we follow the heuristic of adding the subject with the same argument label that it has in the first conjunct, unless the second verb already has that argument, in which case we do not add anything. Neither do we add modals or negations if the second verb already has them.

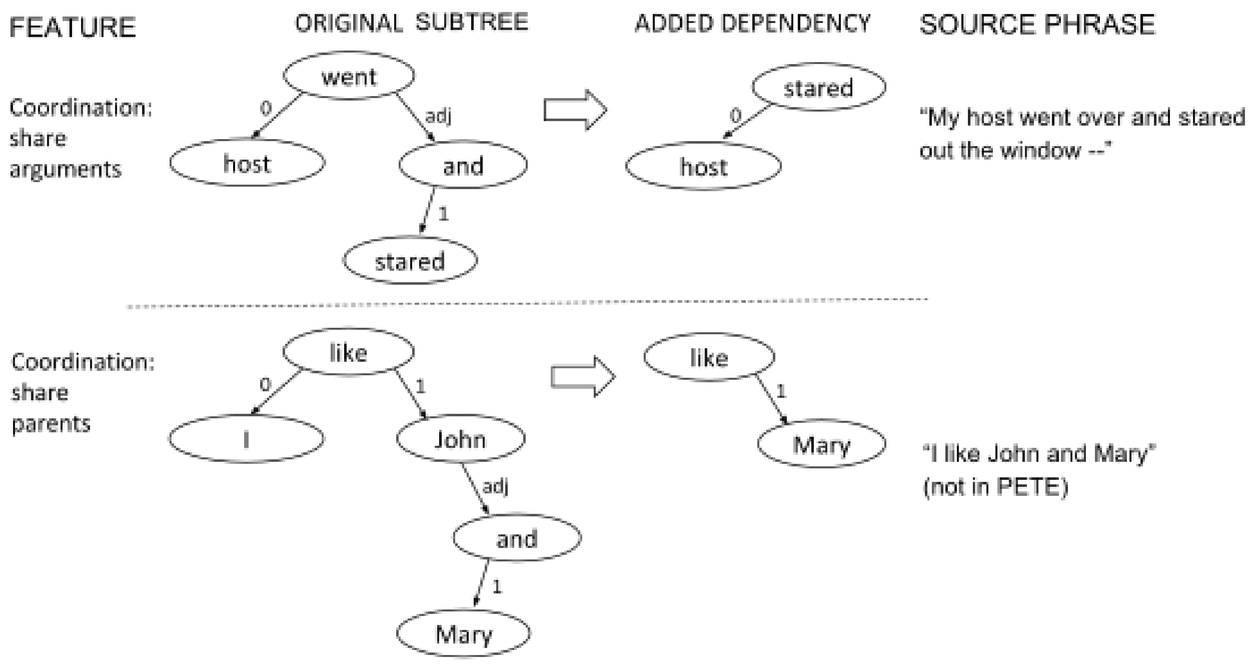
FEATURE
ORIGINAL SUBTREE
ADDED DEPENDENCY
SOURCE PHRASE
Coordination: share arguments
went
0
adj
host
and
1
stared
stared
0
host
"My host went over and stared out the window --"
Coordination: share parents
like
0
1
I
John
adj
and
1
Mary
like
1
Mary
"I like John and Mary" (not in PETE)

Figure 2: Structural transformations for coordination.

In Section 6.2, we report our main results based on the transformations and this notion of subderivation above, and compare those results to other systems. It is interesting to observe that the Cambridge system (Rimell and Clark, 2010) made use of the following heuristic procedure to determine whether the result of one CCG derivations (converted into a dependency structure) entails another:

1. Lowercase and lemmatize all tokens

2. From D_H, discard dependencies involving tokens not present in D_T.

3. Let core(D_H) be the dependencies with subject and object relations in D_H. Answer YES if $core(D_H) \subset core(D_T)$ and $D_H \cap D_T = \emptyset$, NO otherwise.

We see little a priori motivation for the wholesale elimination of tokens in H not present in T, or for the restriction to subject and object relations. Nonetheless, for comparability, we also include separate results where we apply our structural transformations, but rather than using our notion of subderivation, instead follow the heuristic procedure adopted in the Cambridge system.

4 Supertagging Models

In the experiments reported below, we compare performance with two different supertaggers.

4.1 bi-LSTM Supertagger

The current state-of-the-art in TAG supertagging is reported in (Kasai et al., 2017), a model based on a bidirectional LSTM that get as input word sequences and predicted part of speech tags, and produces as output a probability distributions over the TAG supertags at the last softmax layer. This supertagger is trained on Section 1-22 in the TAG-annotated WSJ Penn Tree Bank extracted by Chen (2001). Its supertagging accuracy on Section 0 is 89.32%. For more details, see Kasai et al. (2017).

4.2 MICA Supertagger

The MICA supertagger is a maxent model, which uses lexical and part-of-speech attributes of words in a 3-word window on either side of the target word in a one-versus-all classification task. (The tagger does not use tagging history.) It achieves 88.52% accuracy on Section 0.

5 Parsing Models

We compare two parsing models here, one a neural network-based transition based parser, and another a chart parser.

5.1 Shift-Reduce Neural Network TAG Parser

The currently best performing TAG parser is reported in Kasai et al. (2017). This is an arc-eager shift-reduce parser that uses a feed-forward neural network as an oracle. At each time step, the oracle takes as input the configuration of the parser, which consists of a fixed number of cells from the top of the stack and the front of the buffer, each containing a 1-best supertag from the supertagger. The parser's task is to construct a derivation tree from the individual supertags. This derivation is constructed in the usual way for transition-based parsers, namely through a series of actions (shift, reduce, left-arc, and right-arc). Left-arc and right-arc create links in the derivation tree, and these operations are further specified by the type of operation (substitution and adjoining) as well as the node within the elementary tree to which the operation applies (specified for substitution as 0-4, encodings of the deep grammatical role of the substitution site). The output of the network is a softmax layer, whose activations can be interpreted as a probability distribution over actions and labels. It should be noted that the parser is unlexicalized; the only information that the parser uses to determine its action is the supertags in the relevant cells of the stack and buffer. We train this parser on WSJ Sections 1-22 similarly to the bi-LSTM supertagger. For more details, see Kasai et al. (2017).

5.2 MICA Parser

The MICA parser (Bangalore et al., 2009) is based on the SYNTAX system (Boullier, 2003), a full Earley parser with additional performance optimizations to deal with large grammars. The TAG grammar is transformed into a variant of a probabilistic CFG which allows Kleene stars on righthand side nonterminals to model adjunction. MICA produces a full parse forest and can output n-best parses, but in these experiments we only consider the 1-best parse.

5.3 Intrinsic task results

Table 1 shows that the NN parser outperforms the MICA parser on both the development set (Section 0) and the test set (Section 23).

6 Results and Discussions

The results of PETE are provided using both accuracy and f-measure for finding the entailment cases (which represent around 52% of the cases in the test set). Both metrics are reasonable metrics for the task, and the choice of primary metric depends on exactly why one wants to perform this extrinsic textual entailment task. The fact that both metrics are reasonable ways of assessing performance makes it more difficult to rank the systems, and we comment on both accuracy and f-measure.

6.1 Previous Results

The top part of Table 2 shows the previous results from the best performing systems in the 2010 SEMEVAL PETE task. All of these involve systems which make use of grammatical formalisms that provide rich linguistic description: the CCG-based Cambridge and SCHWA systems, and the HPSG/MRS-based system from Lien (2014). The Cambridge system does best by accuracy, the SCHWA system by f-measure. We note that the MRS system, which uses a grammar to derive a deeper semantics from the parse than we are using in our approach, gets the highest precision among these systems, but at the cost of a lower recall, as is often the case with grammar-based systems.

6.2 Performance

The middle part of Table 2 shows results using structural transformations and our notion of subderivation. We evaluate our system with three combinations of MICA and neural network supertaggers and parsers. In general, the more neural networks, the better. We note a large discrepancy between our results for development and test set (accuracies 78.8% v.s 68.4% for bi-LSTM+NN system). We note that all of our transformations and subderivation notion *increase* the number of yes-answers (by enlarging T or reducing H). Yet, our high precision and low recall values show that we miss a significant amount of 'yes'-answers. Thus, we evidently did not devise enough transformations; or perhaps our formulation of the criterion for entailment was not applicable to the test set.

6.3 Cambridge Heuristics

Because of the drop off when using our subderivation notion, we also explored the use of the Cambridge heuristics. The bottom part of Table 2

		Dev Results				Test Results			
		Gold Stags		Predicted Stags		Gold Stags		Predicted Stags	
Parsing Model	Beam size	UAS	LAS	UAS	LAS	UAS	LAS	UAS	LAS
MICA Stagger + MICA Parser	–	97.60	97.30	87.91	86.14	96.97	96.59	86.66	84.90
bi-LSTM Stagger + MICA Parser	–			90.05	88.32			90.20	88.66
bi-LSTM Stagger + NN Parser	1	96.82	96.45	89.48	88.00	–	–	–	–
bi-LSTM Stagger + NN Parser	16	**97.67**	**97.45**	**90.23**	**88.77**	**97.87**	**97.64**	**90.25**	**88.91**

Table 1: Intrinsic task parsing results on the development and test sets.

		Entailment	Dev Results				Test Results			
Supertagger	Parser	Identification	%A	%P	%R	%F1	%A	%P	%R	%F1
Cambridge	Cambridge	Cambridge	66.7	78.6	57.9	66.7	**72.4**	79.6	62.8	70.2
SCHWA	SCHWA	SCHWA	78.8	84.2	**80.0**	**82.0**	70.4	68.3	**80.1**	**73.7**
MRS	MRS	MRS	77.3	92.6	65.8	76.9	70.7	88.6	50.0	63.9
bi-LSTM	Feed-Forward NN	Subderivation	**78.8**	**92.9**	68.4	78.8	68.4	90.7	43.6	58.9
bi-LSTM	MICA	Subderivation	74.2	92.0	60.5	73.0	66.1	**90.9**	38.5	54.1
MICA	MICA	Subderivation	68.2	90.5	50.0	64.4	63.8	87.3	35.3	50.2
bi-LSTM	Feed-Forward NN	Cambridge	66.7	73.5	65.8	69.4	72.4	85.4	56.4	68
bi-LSTM	MICA	Cambridge	69.7	75	71.1	73	**75.7**	88.1	61.5	72.5

Table 2: PETE task previous system scores and TAG system scores on dev and test sets, using structural transformations together with either our notion of subderivation or Cambridge's heuristics. Accuracy (A) gives the percentage of correct answers for both YES and NO. Precision (P), recall (R) and F1 are calculated for YES.

shows results using our structural transformations together with Cambridge heuristics for entailment identification.

Surprisingly, the system comprised of the neural supertagger and MICA parser is the best performing of our systems now, based on both accuracy and f-measure. In fact, it has the best result on accuracy compared to all systems, and is beat only by SCHWA in f-measure. We note that the development set results with the Cambridge heuristics are much lower than with our subderivation notion. This shows that the heuristics are crucial in this evaluation, and that they apply differently for the development and test sets.

6.4 Discussion

We note a disconnect from parser performances when we use the Cambridge heuristic. We can compare these results to the intrinsic evaluation results of Table 1. The bi-LSTM+NN model performs better on the intrinsic parsing evaluation metrics UAS and LAS (unlabeled and labeled attachment scores) than the bi-LSTM+MICA model. This is reflected in the results using structural transformations and our subderivation notion. However, this performance difference is inverted with structural transformations and Cambridge heuristics.

These results also document that the heuristics are crucial in this evaluation; the Cambridge heuristics do not appear to be motivated by general linguistic considerations. It appears to be important to understand the specific data chosen for the evaluation in order to derive good heuristics. As noted in Section 2, the T sentences were manually selected and transformed by the workshop organizers, in order to maximize the presence of certain syntactic phenomena known to be difficult. The entailment sentences (H) were constructed by hand using certain types of transformations, which are not completely specified in the task description. So there is a certain human idea (that of the workshop organizers) of "entailment" operating here which we are trying to reverse engineer. As we created our heuristics based only on the development set and kept to linguistically well-grounded notions, we ended up with low recall on the test set. Consequently, a good understanding of the data creation process will probably benefit performance. This is not to say PETE is not an interesting evaluation exercise, but it is (necessarily) limited in what it shows because of the data creation process.

6.5 Error analysis

6.5.1 Sources of Errors

In order to better understand the nature of the data set as well as what kinds of cases are causing problems for the neural network TAG parser, we inspected the examples from the development set that led to errors. A number of these were parsing errors without any simple diagnosis. However a number of the others were more intriguing.

One case in the development set was misparsed because of an attachment ambiguity: *It is the last of the three tests of manhood which the women impose* is supposed to entail that *the women impose tests*. The NN parser mistakenly attaches the relative clause headed by *impose* to the noun *manhood* instead of to *tests*. Such examples inevitably pose difficulties for our TAG-based parsers, since they are unlexicalized and therefore cannot make use of lexical information to make attachment decisions.

Another case in the test set involves the resolution of an anaphoric dependency: *Mary said 'I have seen'* is supposed to entail *Mary has seen something*. It is not clear to us that such cases should be treated as syntactically governed (or even if the entailment is correct – she could be mistaken or lying). If the verb is changed from *said* to *heard*, the entailment no longer goes through, suggesting that the lexical semantics of the embedding predicate, not to mention the indexical, is at issue.[3]

The fine-grained character of TAG derivations gives rise to derivational ambiguities that do not exist in other frameworks, but which can prevent the recognition of entailments. The development set contains the phrase *Japan hadn't come up with specific changes*. For reasons that are not clear to us at present, the phrasal verb is parsed correctly in H, with both *up* and *with* treated as co-heads to the verb *come*. However in the parse of T, which also contains this phrasal verb, only *up* is treated as a co-head, while *with* is treated as the head of a PP modifying the VP. One of the major differences between the Cambridge heuristics and our subderivation notion is the former's exclusive focus on "core arguments". This would allow misparses of this sort to be ignored, though it might pose problems for other cases.

Finally, we note a surprising parsing error arising from an error in Part of Speech tagging that is input to our supertagger. For the Hypothesis *Many bear resemblances to movie personalities*, both our bi-LSTM supertagger and the Stanford PCFG Parser (Manning et al., 2014) tag *many bear resemblances* as an adjective, noun, and verb (as in *red bear runs*), instead of the correct sequence determiner, verb, and noun. In the pre-trained Parsey McParseface model (Andor et al., 2016), *many bear resemblances* is tagged as an adjective, noun, and noun, a sequence that is locally possible, but not compatible with the sentence as a whole. This is surprising, given that POS tagging is often regarded as a largely solved task. This sentence is short and contains no unbounded dependencies, nor are its lexical items unusual.

6.5.2 Differences between Subderivation Condition and Cambridge Heuristics

To compare our subderivation condition to the Cambridge heuristics, we compared their respective test set results under the bi-LSTM+NN model: out of 301 entailment hypotheses, there are 20 cases that are correctly recognized by Cambridge heuristics but not our heuristics, and eight cases for which the opposite is true.

All of the 20 cases that Cambridge heuristics succeed in are True entailments that our heuristics mislabeled as False. Eight cases involve parser errors: either one of two coheads was mislabeled as an adjunction, a relative clause was misattached due to an attachment ambiguity, the grammatical function of the relative clause head is misidentified, or quotation marks are misparsed. Eight cases stemmed from the fact that our subderivation condition does not involve lemmatization of non-auxiliary verbs; hence it will fail on cases where a non-auxiliary verb is in different forms in the hypothesis and entailment, for example *Something includes* vs. *Examples include*. One further failure involved ellipsis, whose underlying structure our TAG parse does not allow us to recover: with the text *Consider the ingredients, not the name*, our parse attaches *not* to *name*, whereas with the hypothesis *Do not consider the name*, our parse attaches *not* to *consider*. Another case involved a difference in choice of determiner between the hypothesis and text, i.e., *the* vs. *an*. A final case involved what we take to be a contestable gold label, where the text *he would have a place to hang* is granted the hypothesis *he would hang* as a True entailment.

For these cases, the Cambridge heuristics succeed by lemmatizing all tokens (8 cases), by considering only the core arguments (11 cases), and by skipping an unseen token (1 case).

In contrast, the eight cases where Cambridge heuristics fail but where the subderivation condi-

[3]Rimell and Clark (2010) also cite the example discussed earlier, *trading is something we want to watch*, as anaphoric. However, we believe that a syntactic treatment can be given if predicative sentences are correctly recognized.

tion succeeds are all instances of False entailments that Cambridge heuristics mislabels as True. Here, we see that the Cambridge heuristics lose relevant grammatical distinctions as a result of ignoring non-core arguments (6 cases) or skipping unseen tokens (2). For example, with the text *that is exactly what I'm hoping for* and hypothesis *I'm hoping exactly*, it is the non-core adjunction relation between *exactly* and *hoping* that makes the hypothesis False. Or, with the text *to live like Christians* and hypothesis *someone likes Christians*, it is the unseen token *someone* with the tokens *like* or *likes* that make the hypothesis False.

As observed in Table 2, precision is higher but accuracy is lower for our subderivation condition than for Cambridge heuristics under the bi-LSTM+NN model. Based on the inspection above, Cambridge heuristics result in gains and losses in entailment prediction as a result of their coarse-grained nature. Our subderivation condition fails to predict certain entailments because of its lack of verb lemmatization and its lack of robustness to "minor" parse failures.[4] However, its success is more directly tied to parser performance in a more linguistically rigorous way.

7 Conclusions and Future Work

In this paper, we have presented results for the PETE task using three systems for TAG parsing. Our results confirm previous exploration of this task in which parsers that provide rich linguistic descriptions fare best. Using linguistically motivated structural transformations and a subderivation criterion for detecting entailments, we have shown that TAG parsers can outperform the state-of-the-art on the development sets. However, on the test set, our results decrease sharply. In contrast, when we instead apply the heuristics used by the (previously) best performing CCG-based system (Cambridge) to detect entailment, we obtain accuracy results that surpass the state-of-the-art on this task. These results demonstrate that heuristics greatly affect task performance.

Since the development set is small, and we inferred structural transformations by observing the linguistic changes in the development set, there may be linguistic features that TAG parses do provide, but which we did not think to make use of. This reflects a common challenge for rule-based systems. More rules could be derived by observing the Penn Treebank and our particular grammar. We intend to analyze which cases the Cambridge heuristics perform better on than our subderivation-based heuristics; in so doing, we will treat the test set as an expanded development set, but we feel the provided development set is too small to truly understand this problem. We hope it will be possible to create a new test set in the future if there is sufficient community interest.

Finally, because our structural transformations are based on general properties of TAG derivations and are task-independent, this suggests that the utility of the current work may extend to other extrinsic tasks, such as semantic role labeling and textual entailment tasks that involve lexical semantics, world knowledge and logical reasoning. We leave such explorations for the future.

[4]Note that if we add lemmatization of main verbs to our subderivation condition, we gain 8 additional correct predictions (the aforementioned lemmatization failures), but lose no other cases. This would yield the following performance scores: accuracy 71.1, precision 91.6, recall 48.7, and F1 63.6.

References

Bharat Ram Ambati, Tejaswini Deoskar, and Mark Steedman. 2016. Shift-Reduce CCG Parsing using Neural Network Models. In *Proceedings of NAACL-HLT 2016*. pages 447–453.

Daniel Andor, Chris Alberti, David Weiss, Aliaksei Severyn, Alessandro Presta, Kuzman Ganchev, Slav Petrov, and Michael Collins. 2016. Globally Normalized Transition-Based Neural Networks. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*. pages 2442–2452.

Srinivas Bangalore, Pierre Boullier, Alexis Nasr, Owen Rambow, and Benoît Sagot. 2009. MICA: A Probabilistic Dependency Parser Based on Tree Insertion Grammars. In *NAACL HLT 2009 (Short Papers)*.

Steven Bird, Ewan Klein, and Edward Loper. 2009. *Natural Language Processing with Python*. OReilly Media.

Pierre Boullier. 2003. Guided Earley parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT03)*. Nancy, France, pages 43–54.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine n-best parsing and maxent discriminative reranking. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*. pages 173–180.

Danqi Chen and Christopher D Manning. 2014. A Fast and Accurate Dependency Parser using Neural Networks. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*.

John Chen. 2001. *Towards efficient statistical parsing using lexicalized grammatical information*. Ph.D. thesis, University of Delaware.

Stephen Clark and James R. Curran. 2007. Wide-coverage efficient statistical parsing with CCG and log-linear models. *Computational Linguistics* 33(4).

Timothy Dozat and Christopher D. Manning. 2017. Deep biaffine attention for neural dependency parsing. In *Proceedings of ICLR*. http://arxiv.org/abs/1611.01734.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A Smith. 2015. Transition-based Dependency Parsing with Stack Long Short-term Memory. In *Proceedings of the conference on Empirical Methods in Natural Language Processing (EMNLP)*. pages 334–343.

Chris Dyer, Adhiguna Kuncoro, Miguel Ballesteros, and Noah A. Smith. 2017. Recurrent Neural Network Grammars. In *Proceedings of NAACL*. pages 1249–1258.

Jungo Kasai, Robert Frank, R. Thomas McCoy, Owen Rambow, and Alexis Nasr. 2017. Tag parsing with neural networks and vector representations of supertags. In *Proceedings of EMNLP*. Association for Computational Linguistics.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, Graham Neubig, and Noah A. Smith. 2017. What Do Recurrent Neural Network Grammars Learn About Syntax? In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics (EACL)*. pages 1249–1258.

Mike Lewis, Kenton Lee, and Luke Zettlemoyer. 2016. LSTM CCG Parsing. In *Proceedings of NAACL-HLT 2016*. pages 221–231.

Elisabeth Lien. 2014. Using minimal recursion semantics for entailment recognition. In *Proceedings of the Student Research Workshop at the 14th Conference of the European Chapter of the Association for Computational Linguistics*. Gothenburg, Sweden, page 7684.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *Association for Computational Linguistics (ACL) System Demonstrations*. pages 55–60. http://www.aclweb.org/anthology/P/P14/P14-5010.

Dominick Ng, James W.D. Constable, Matthew Honnibal, and James R. Curran. 2010. SCHWA: PETE using CCG dependencies with the C&C parser. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. page 313316.

Laura Rimell and Stephen Clark. 2010. Cambridge: Parser evaluation using textual entailment by grammatical relation comparison. In *Proceedings of the 5th International Workshop on Semantic Evaluation*. pages 268–271.

Laura Rimell, Stephen Clark, and Mark Steedman. 2009. Unbounded dependency recovery for parser evaluation. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing*. Singapore, page 813821.

XTAG Research Group. 2001. A lexicalized tree adjoining grammar for English. Technical Report IRCS-01-03, Institute for Research in Cognitive Science, University of Pennsylvania.

Wenduan Xu, Michael Auli, and Stephen Clark. 2015. CCG supertagging with a recurrent neural network. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Short Papers)*. pages 250–255.

Deniz Yuret, Laura Rimell, and Aydin Han. 2013. Parser Evaluation Using Textual Entailments. *Language Resources and Evaluation* 47(3):639–659.