Stanford's System for Parsing the English Web

David McClosky[s*], **Wanxiang Che**[h*], **Marta Recasens**[s], **Mengqiu Wang**[s], **Richard Socher**[s], **and Christopher D. Manning**[s]
[s] Natural Language Processing Group, Stanford University
{mcclosky,recasens,mengqiu,socherr,manning}@stanford.edu
[h] School of Computer Science and Technology, Harbin Institute of Technology
car@ir.hit.edu.cn

Abstract

We describe the Stanford entries to the SANCL 2012 shared task on parsing non-canonical language. Stanford submitted three entries: (i) a self-trained generative constituency parser, (ii) a graph-based dependency parser, and (iii) a stacked dependency parser using the output from the constituency parser as features while parsing. The stacked parser obtained 2nd place in the dependency parsing track. Our overall approach involved exploring techniques which improved performance consistently across domains without using many external resources.

1 Introduction

Extracting the syntactic structure of non-canonical language (such as emails and blog posts) is a task important for gaining a more complete understanding of humanity's digital discourse. This was the focus of the SANCL (Syntactic Analysis of Non-Canonical Language) shared task by Petrov and McDonald (2012).

In this paper, we describe Stanford's three entries to SANCL 2012. The first entry (§2) is a semi-supervised generative constituency parser (McClosky et al., 2006). The second entry (§3) is a graph-based dependency parser along the lines of McDonald et al. (2005b). Our third entry (§4) combines the first two by using the output from the constituency parser as stacking features in dependency parser.

2 Constituency parsing

Our constituency parser is the BLLIP reranking parser[1] (Charniak and Johnson, 2005). The BLLIP parser consists of a PCFG-like constituency parser (BLLIP_1) and a discriminative reranker (BLLIP_2). This results in two possible parsing models.

To take advantage of the unlabeled corpora, we apply self-training (McClosky et al., 2006) in the following way. First, we train the parser and reranker on the OntoNotes WSJ labeled data. Next, using that reranking parser we parse all five unlabeled corpora. For each unlabeled corpus, we train a parsing model. Finally, we build a new "uniform" parsing model by linearly interpolating the counts from the parsing models trained on OntoNotes WSJ and the parsed unlabeled corpora. We ensure that each corpus gives an equal contribution accounting for the relative size of each corpus. For example, given two corpora where one has ten times as many words, we would give the larger corpus a weight of 0.1 and the smaller corpus a weight of 0.9. The results of this can be seen in Table 1. The uniform self-trained models perform better across the three development domains. Additional results can be found at http://mlcomp.org.

To parse the unlabeled text during self-training or at test time, we use the -s (small corpus) flags. The latter ensures that all edges receive positive merit even if the specific construction has not been seen in the training data. We found that this improved performance on development data.

We experimented with customizing the parsing

[*] These authors contributed equally.

[1] http://github.com/BLLIP/bllip-parser/

Parser	Model	Evaluation set (dev) WSJ	Emails	Weblogs
$BLLIP_1$	WSJ	89.3	78.3	83.1
$BLLIP_2$	WSJ	**91.2**	79.4	85.4
$BLLIP_1$	Uniform	89.5	80.9	85.3
$BLLIP_2$	Uniform	90.9	**81.1**	85.6
$BLLIP_2$	Adapted	**91.2**	**81.1**	**85.7**
$BLLIP_2$	Oracle	91.3	81.5	85.9

Table 1: F_1 scores BLLIP parsers on the development sections of three domains. The "Uniform" model is self-trained on all five unlabeled corpora. "Adapted" refers to using automatic domain adaptation, averaged over 10 folds. "Oracle" picks the best parsing model from 400 random combinations of parsing models for each evaluation set. This represents the limit of the automatic domain adaptation method.

models to the target text as in McClosky et al. (2010). We found that for this set of source and target texts (one labeled corpus and five unlabeled corpora), automatic domain adaptation performed about as well as the uniform baseline. We believe that this may stem from only having a single labeled corpus which made it difficult to learn useful cross-domain divergence measures. Given its additional complexity, our *Stanford* and *Stanford-1* submissions use the "Uniform" model.

3 Dependency parsing

Our baseline dependency parser (*Stanford-2*) adopted the state-of-the-art graph-based dependency parsing (Kübler et al., 2009). The score of a dependency tree is factored into scores of small parts (sub-trees) and the graph-based dependency parsing views the problem of finding optimal dependency tree as finding the maximum spanning tree (MST) from a directed graph. Based on dynamic programming decoding, it can find efficiently an optimal tree in a huge search space.

Here we used Mate parser[2] (Bohnet, 2010), an open source implementation of Carreras (2007)'s second-order MST dependency parsing algorithm. Besides the first-order features (McDonald et al., 2005a) and second-order features[3] (McDonald and Pereira, 2006), Carreras (2007) also included any sibling and grandchild occurring in the sentence between the head and the modifier. In addition, relation label prediction is an integral part of the algorithm. Unlike Carreras (2007), the passive-aggressive perceptron algorithm (Crammer et al., 2006) was used to learn feature weights in the Mate parser. To reduce the number of loops over all kinds of relation labels, only those labels that were present in the training corpus for a head and modifier POS combination were considered (Johansson and Nugues, 2008). In order to speed up the feature extraction process, we employed hash kernels.[4] Although the conflicts introduced by hash kernel can harm parsing accuracy, the technique dramatically reduces the feature space and speeds up the feature extraction process. Additionally, it makes it possible to take into account the features over negative examples during the training stage, which significantly improves the parsing performance. Furthermore, the Mate parser can take advantage of a modern multi-core CPUs to extract features in parallel. Therefore, the Mate parser not only achieves high accuracy, but also high efficiency at training and testing time.

Most dependency parsers require exogenous part-of-speech (POS) tags. To obtain these, we used the Stanford tagger (Toutanova et al., 2003) with the `bidirectional-distsim-wsj-0-18.tagger` model.[5] The model was trained on WSJ sections 0–18 and an extra POS tagging corpus (18,589 sentences), using a bidirectional architecture and including word shape and distributional similarity features. We experimented by retraining the tagger on the training section of OntoNotes WSJ but obtained worse performance on the Emails and Weblogs domains. This may be because the original Stanford Tagger model uses a small amount of additional training data and includes distributional similarity features. The evaluation on the test sets shows that although the POS tagger obtained low in-domain scores (OntoNotes WSJ)—the last place but one in 12 submission systems—it achieved high out-of-domain scores (4th place).

Word lemmas are another feature known to be useful for dependency parsing. We obtained the

[2] `code.google.com/p/mate-tools`

[3] McDonald and Pereira (2006)'s second-order features only included sibling nodes which are closest to the modifier. In addition, they used a separate algorithm to predict relation labels.

[4] Hash kernels directly map a string feature into an integer according to a hash function, allowing for conflicts.

[5] `nlp.stanford.edu/software/tagger.shtml`

lemma of each word with the `Morphology` class of the Stanford JavaNLP tool.[6] The Mate parser was trained over 6 iterations. This number was tuned on development data.

4 Stacked system

Stacked learning (Wolpert, 1992) is a general framework in which a predictor is trained to improve the performance of another and the two predictors are complementary. The method has been successfully applied to dependency parsing, where two dependency parsing models—a graph-based and a transition-based— can help each other (Nivre and McDonald, 2008; Torres Martins et al., 2008). Inspired by their work, we propose a stacked learning method (*Stanford-1*) that leverages constituent parsing results (*Stanford*) in the graph-based dependency parser (*Stanford-2*).

During training, we obtained parses of the training data by running the BLLIP_1 parser in 20-fold cross-validation and then converted the constituency parses into Stanford Dependencies.[7] Next, we extracted stacked learning features from the output dependency tree for Mate parser as additional features. Development and test data were parsed using BLLIP_2 trained on all training data. Table 2 shows the stacked learning features used in our system. The stacked system uses the same POS tags as the dependency parser (*Stanford-2*).

5 Spelling and acronym experiments

We experimented with applying a set of high-precision text replacements to the unlabeled data and target texts. These consisted of 1,057 spelling autocorrection rules (e.g. "yuo" → "you") from Pidgin instant messaging client[8] along with 151 common Internet abbreviations (e.g. "LOL" → "laughing out loud"). Our hope was that these corrections would reduce the number of unknown words (many of which are mistagged) and provide a better corpus for self-training. Unfortunately, we found that spelling errors are actually fairly infrequent and these spelling corrections only changed a small percentage of sentences in the unlabeled text (ranging from under 1% in Weblogs to 9.9% in Answers). Roughly half of the corrections involved uppercasing the word "i" which does not change the parse structure in the BLLIP parser.[9] As a result, the potential gain for this approach is rather limited.

Dependency
$in((h,m,*),\mathbf{d}^C)$
$in((h,m,l),\mathbf{d}^C)$
$in((h,m,*),\mathbf{d}^C)\circ t_h\circ t_m$
$in((h,m,l),\mathbf{d}^C)\circ t_h\circ t_m$
Sibling
$in((h,m,*),\mathbf{d}^C)\circ in((h,s,*),\mathbf{d}^C$
$in((h,m,*),\mathbf{d}^C)\circ in((h,s,*),\mathbf{d}^C)\circ t_h\circ t_m\circ t_s$
$in((h,m,l),\mathbf{d}^C)\circ in((h,s,l),\mathbf{d}^C)$
$in((h,m,l),\mathbf{d}^C)\circ in((h,s,l),\mathbf{d}^C)\circ t_h\circ t_m\circ t_s$
Grandchild
$in((h,m,*),\mathbf{d}^C)\circ in((m,g,*),\mathbf{d}^C)$
$in((h,m,*),\mathbf{d}^C)\circ in((m,g,*),\mathbf{d}^C)\circ t_h\circ t_m\circ t_g$
$in((h,m,l),\mathbf{d}^C)\circ in((m,g,l),\mathbf{d}^C)$
$in((h,m,l),\mathbf{d}^C)\circ in((m,g,l),\mathbf{d}^C)\circ t_h\circ t_m\circ t_g$

Table 2: Stacked learning features. $in(d,\mathbf{d})$ is an indicator function, which is 1 if the dependency arc $d=(i,j,l)$ is in dependency tree $\mathbf{d}$, where *head* word (or *father*) is at position i and *modifier* (*dependent* or *child*) is at j, with a dependency relation label l. $*$ represents any relation. h,m,s,g denote positions of head, modifier, sibling, and grandchild respectively. t is the POS tag. $\mathbf{d}^C$ is the dependency tree from the BLLIP_1 parser. $\circ$ is the feature conjoining operator.

6 Results and discussion

Table 3 shows the F_1 scores of Mate, BLLIP_1, BLLIP_2, and stacked learning parsers for the most frequent relation labels on OntoNotes WSJ development data. In these experiments, all systems are trained only on OntoNotes WSJ training data. The stacked dependency parser performs better for the majority of relations. A comparison between Mate and BLLIP_2 shows that the former achieves higher scores for some relations, noun compound modifiers (*nn*), adjectival modifiers (*amod*) and direct objects (*dobj*) in particular; whereas the latter obtains higher scores for *root*, conjunct (*conj*), depen-

[6] `nlp.stanford.edu/software/corenlp.shtml`

[7] During training, we use BLLIP_1 instead of BLLIP_2, because using BLLIP_2 would require training a reranker for each fold which in turn requires cross-validation.

[8] `http://pidgin.im/`

[9] This is because the unknown word model in the Charniak parser only considers the case of a token if its caseless form is novel.

Relation	Count	Mate	BLLIP_1	BLLIP_2	Stack
prep	2,912	97.5	97.5	98.0	**98.1**
pobj	2,825	96.1	95.5	96.3	**96.5**
nn	2,811	95.1	94.0	94.0	**95.5**
det	2,644	**99.3**	99.2	**99.3**	**99.3**
nsubj	2,215	95.7	95.5	96.2	**96.4**
amod	1,809	93.5	92.5	92.5	**94.1**
root	1,335	95.7	95.9	**96.3**	96.1
dobj	1,231	93.7	91.9	92.8	**94.5**
advmod	1,066	90.7	90.3	90.8	**91.6**
aux	1,007	**98.7**	98.3	98.5	**98.7**
cc	776	**99.4**	99.2	99.2	**99.4**
conj	773	88.3	90.6	**92.2**	91.7
num	732	96.3	95.7	96.3	**96.9**
dep	588	54.3	57.5	60.5	**62.3**
poss	541	98.2	98.0	**98.8**	98.5

Table 3: Performance (F_1 scores) of Mate, BLLIP_1, BLLIP_2, and stacked learning parsers for the fifteen most-frequent dependency relations on the OntoNotes WSJ development dataset.

dent (*dep*), nominal subjects (*nsubj*), prepositional modifiers (*prep*), and possession modifiers (*poss*).

The following example illustrates the complementarity of the two parsers, from which the stacked system benefits. Mate, unlike BLLIP_2, gets entirely wrong the coordination structure for "KENNEDY, SOUTER, and GINSBURG joined", involving *nn*, *nsubj*, *cc*, and *conj*. With input from BLLIP_2, the stacked parser yields the correct parse of the sentence:

> BREYER filed a concurring opinion, in which KENNEDY, SOUTER, and GINSBURG joined.

Mate seems to excel at short-distance dependencies, possibly because it uses more local features (even with a second-order model) than BLLIP, whose PCFG and reranking strategy can capture long-distance dependencies. This suggests a plausible reason for the higher scores achieved by the combined approach, stacked learning. For the *conj* relation, BLLIP_2 obtains the best performance presumably since it directly models coordination (e.g., the Coordination Parallelism feature in the reranker).

The final test results of our dependency parsing systems on the test datasets are shown in Table 4, where the output of our constituency parser (*Stanford*) was converted into dependency parse trees.

System	WSJ domain		A-C domains	
	LAS	UAS	LAS	UAS
Stanford (BLLIP_2)	90.3	92.5	82.8	86.9
Stanford-2 (Mate)	89.9	92.0	80.3	84.7
Stanford-1 (Stack)	91.5	93.4	83.1	87.2

Table 4: Performance of different systems on test.

The stacked parser (*Stanford-1*) obtained 2nd place in the dependency parsing track.

7 Conclusion

We described Stanford's three entries in the SANCL shared task. Using stacking, we combined a self-trained generative constituency parser with a graph-based dependency parser and obtained 2nd place in the dependencies track. The only external resource used was the Stanford POS tagger.

There are many possible directions for future work. These include obtaining a better understanding of what circumstances allow automatic domain adaptation to perform well and determining better methods of combining the source domains to produce more robust parsing models. For future instances of this shared task, we feel it would be useful to include multiple labeled domains for training to facilitate additional forms of semi-supervised domain adaptation.

Acknowledgements

DM, MW, RS, and CM gratefully acknowledge the support of Defense Advanced Research Projects Agency (DARPA) Machine Reading Program under Air Force Research Laboratory (AFRL) prime contract no. FA8750-09-C-0181 and the support of the DARPA Broad Operational Language Translation (BOLT) program through IBM. Any opinions, findings, and conclusion or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the view of the DARPA, AFRL, or the US government.

WC was supported in part by the National Natural Science Foundation of China (NSFC) via grant 61133012, the National "863" Project grant 2011AA01A207 and 2012AA011102.

MR was supported in part by a Beatriu de Pinós postdoctoral scholarship (2010 BP-A 00149) from Generalitat de Catalunya.

References

Bernd Bohnet. 2010. Top accuracy and fast dependency parsing is not a contradiction. In *Proceedings of Coling 2010*, pages 89–97.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proceedings of the CoNLL Shared Task Session of EMNLP-CoNLL 2007*, pages 957–961.

Eugene Charniak and Mark Johnson. 2005. Coarse-to-fine *n*-best parsing and MaxEnt discriminative reranking. In *Proceedings of ACL 2005*, pages 173–180.

Koby Crammer, Ofer Dekel, Joseph Keshet, Shai Shalev-Shwartz, and Yoram Singer. 2006. Online passive-aggressive algorithms. *Journal of Machine Learning Research*, 7:551–585.

Richard Johansson and Pierre Nugues. 2008. Dependency-based syntactic–semantic analysis with PropBank and NomBank. In *Proceedings of CoNLL 2008*, pages 183–187.

Sandra Kübler, Ryan T. McDonald, and Joakim Nivre. 2009. *Dependency Parsing*. Synthesis Lectures on Human Language Technologies. Morgan & Claypool Publishers.

David McClosky, Eugene Charniak, and Mark Johnson. 2006. Effective self-training for parsing. In *Proceedings of HLT-NAACL 2006*, pages 152–159.

David McClosky, Eugene Charniak, and Mark Johnson. 2010. Automatic domain adaptation for parsing. In *Proceedings of HLT-NAACL 2010*, pages 28–36.

Ryan McDonald and Fernando Pereira. 2006. Online learning of approximate dependency parsing algorithms. In *Proceedings of EACL 2006*, pages 81–88.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proceedings of ACL 2005*, pages 91–98.

Ryan T. McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajic. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proceedings of HLT-EMNLP 2005*, pages 523–530.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proceedings of HLT-ACL 2008*, pages 950–958.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

André Filipe Torres Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proceedings of EMNLP 2008*, pages 157–166.

Kristina Toutanova, Dan Klein, Christopher D. Manning, and Yoram Singer. 2003. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of HLT-NAACL 2003*, pages 173–180.

David H. Wolpert. 1992. Stacked generalization. *Neural Networks*, 5:241–259.