A Gold Standard Dependency Corpus for English

Natalia Silveira[◦], Timothy Dozat[◦], Marie-Catherine de Marneffe[*], Samuel R. Bowman[◦], Miriam Connor[◦], John Bauer[†] and Christopher D. Manning[◦†]

[◦]Linguistics Department, Stanford University, Stanford, CA 94305
{natalias,tdozat,sbowman}@stanford.edu, miriam.connor@gmail.com
[*]Linguistics Department, The Ohio State University, Columbus, OH 43210
mcdm@ling.osu.edu
[†]Computer Science Department, Stanford University, Stanford, CA 94305
{horatio,manning}@stanford.edu

Abstract

We present a gold standard annotation of syntactic dependencies in the English Web Treebank corpus using the Stanford Dependencies standard. This resource addresses the lack of a gold standard dependency treebank for English, as well as the limited availability of gold standard syntactic annotations for informal genres of English text. We also present experiments on the use of this resource, both for training dependency parsers and for evaluating dependency parsers like the one included as part of the Stanford Parser. We show that training a dependency parser on a mix of newswire and web data improves performance on that type of data without greatly hurting performance on newswire text, and therefore gold standard annotations for non-canonical text can be valuable for parsing in general. Furthermore, the systematic annotation effort has informed both the SD formalism and its implementation in the Stanford Parser's dependency converter. In response to the challenges encountered by annotators in the EWT corpus, we revised and extended the Stanford Dependencies standard, and improved the Stanford Parser's dependency converter.

Keywords: dependency grammar, Stanford Dependencies, web treebank

1. Introduction

In this paper, we report on gold standard annotation of syntactic dependencies in the English Web Treebank corpus (Linguistic Data Consortium release LDC2012T13, henceforth EWT), using the Stanford Dependencies (SD) standard (de Marneffe et al., 2006). This resource addresses two major issues in current parsing research: (1) the lack of a gold standard dependency treebank for English; and (2) the limited availability of gold standard syntactic annotations for English informal text genres. We also present initial experiments on the use of this resource, both for training dependency parsers and for evaluating the quality of different versions of the Stanford Parser (Klein and Manning, 2003; de Marneffe et al., 2006).

Almost all parsing tools for English, including the Stanford Parser (Klein and Manning, 2003), are trained on newswire data. This favors performance on a particular type of linguistic data: formal text, written in carefully constructed language and thoroughly revised. The performance of parsers on other tasks then suffers due to this bias. Web data is different: the language is more likely to be non-standard, less formal and more reader-oriented; it is likely to contain more errors and disfluencies. For that reason, applications that target web text (for tasks such as sentiment analysis, information extraction and retrieval, etc.) can benefit from being trained and evaluated on that type of text. Our work is an effort to enable such training and evaluation.

Furthermore, the systematic annotation effort has informed both the SD standard and its implementation in the Stanford Parser's dependency converter (de Marneffe et al., 2006), a tool that produces dependency annotation from constituency trees. In response to the challenges encountered by annotators in the EWT corpus, the formalism has been revised and extended, and the converter has been improved. These changes represent a new phase of data-driven development for both the theoretical and the practical components of the SD formalism, the results of which are discussed here.

2. English Web Treebank

In 2012, the Linguistic Data Consortium (LDC) released the English Web Treebank corpus, consisting of 254,830 word tokens (16,624 sentences) of web text. The text is manually annotated for sentence- and word-level tokenization, as well as part-of-speech tags and constituency structure in the Penn Treebank scheme. The annotation guidelines follow those used in other recent LDC Treebank releases like OntoNotes (Hovy et al., 2006), with richer nominal structure and an augmented tagset for POS. The data comprises five subgenres of web text: blog posts, newsgroup threads, emails, product reviews and answers from question-answer websites.

2.1. Properties of the annotated data

Text on the web differs from more formal registers of English in a number of potentially important ways, motivating the need for training and testing web NLP applications on web data rather than newswire. The distribution of POS-tags and dependency types in the EWT and the WSJ are similar, but close examination reveals clear, quantifiable differences. This section discusses some of the more relevant ones.

Table 1 contrasts the distributions of POS and constituent tags in newswire and web data.[1] In the EWT treebank, there

[1]The two corpora used slightly different label sets; the EWT treebank used 170 unique labels (including functional flags), whereas the OntoNotes WSJ section used 191, with 146 of them overlapping. Disregarding functional flags, EWT used 69 and WSJ used 67, with 63 overlapping.

Tag	EWT	WSJ	Ratio
PRP	4.06E^{-2}	1.56E^{-2}	2.61
SQ*	2.13E^{-3}	2.53E^{-4}	8.43
SBARQ*	8.70E^{-4}	1.61E^{-4}	5.40
FRAG*	5.35E^{-3}	3.97E^{-4}	13.47
NP-VOC	4.10E^{-4}	1.76E^{-5}	23.38
UH	1.99E^{-3}	6.96E^{-5}	28.70
S-IMP	2.79E^{-3}	6.27E^{-6}	445.08
SINV*	5.90E^{-4}	1.28E^{-3}	0.46
QP	1.09E^{-3}	4.27E^{-3}	0.25

Table 1: The frequencies of each tag type in the English Web Treebank and Wall Street Journal data, and the ratios of those frequencies.

are about 463,000 total nodes across all syntactic trees, and in the WSJ section of the OntoNotes corpus, there are over 1.5 million; so the EWT is only 30% the size of the WSJ. Even so, for quite a few important tags, the EWT treebank has significantly more instances than the WSJ.

The percentage of the EWT corpus that comprises PRP labels (about 4%) is over two and a half times greater than the percentage of the WSJ (1.5%), indicating that pronouns (and therefore issues surrounding pronoun resolution) are much more common in web data than in newswire. Questions are likewise more common in web data; the SQ label (including variants with functional tags, such as SQ-PRD), which indicates inversion in an interrogative, and SBARQ, which indicates a main clause *wh*-question, both occur more often in the EWT trees. Discourse phenomena – such as vocatives (NP-VOC) and discourse particles (UH, e.g. *'well'*, *'like'*, and *'please'*) are upwards of 20 times more frequent in the web data. The most striking difference between the two types of data has to do with imperatives, which occur two orders of magnitude more often in the EWT.

Disfluencies and errors are also much more common in web data than in newswire text. Sentence fragments, for instance, are over 13 times more common in EWT.

In contrast, some tags were much less frequent in web data; SINV, which indicates inversion in a declarative sentence, occurred twice as often in the WSJ, probably due to the more formal register, and QP – used with complex number expressions, such as currency – occurred four times as often, almost certainly due to the frequent discussion of finance in the journal. All of these facts reflect the more interactive nature of many web genres, such as email or question-answer forums: the use of pronouns and the frequency of questions reveal a concern with the context of production, and in particular with the addressee.

2.2. Difficult examples

Some concrete examples of the kinds of sentences not found in newswire are given below (1).

(1) a. He's pretty much an "I love American food, good drinks on occasion, laid back" kind of guy. ;)

b. I want to take him somewhere where there's going to be awesome burgers / American food, atmosphere (preferably a tavern / pub style would be nice), good service, and all around a great time.

c. When you have a minute I'll give me a call and I'll come around and show you how to unfreeze the panel.

In (1a), the speaker uses a direct quote as a predicate to describe someone else, which is very informal. The author also ends the sentence with an emoticon, a fairly common practice which is essentially not in newswire text.

At several points in these examples, the author started a phrase one way and ended it another. In (1a), the speaker attempts to conjoin *'laid back'*, an adjectival phrase, with *'American food'* and *'good drinks'* which are nominal. This conjunction violates the constraint in standard text that requires conjoined objects to be matching syntactic types, as well as the constraint that *'love'* take a nominal object. The speaker's construction would be considered ungrammatical in a formal context.

In (1c), there seems to be a restart in the sentence: *'When you have a minute I'll — give me a call and I'll'*. This presents a difficulty for parsing, and is rarely if ever encountered in thoroughly edited text.

3. Dependency Annotation

The corpus was hand-annotated with dependency relations following the SD schema, a *de facto* standard for English syntactic annotation. The result is the first human-checked large-scale gold standard for surface syntax dependency annotation of English text. The Prague Czech-English Dependency Bank (Böhmová et al., 2003) is a similar effort, but in that work the annotation is done automatically, with the exception of some annotations for deep syntax. The BioInfer corpus (Pyysalo et al., 2007) has manual annotation, but, at 33,858 tokens, it is almost eight times smaller than the EWT.

3.1. Stanford Dependencies

The SD representation is widely used in the Natural Language Processing community, in applications such as machine translation or relation extraction. The representation is based on a uniform notation of triples, relating a head word to a dependent word by means of a labeled dependency relation. The standard is designed to be linguistically sophisticated while remaining comprehensible to non-experts, and it preserves some explicit parallels with traditional grammatical formalisms.

3.2. Annotation procedure

Our corpus consists of the hand-corrected output of an automatic parser. To bootstrap the annotation process, we first converted EWT trees to Stanford Dependencies with the Stanford Parser, which outputs dependencies from a Penn Treebank-style parse. The results were then revised, token by token, by specially trained Linguistics Ph.D. students. The unique nature of the data led to extensions and refinements to the Stanford Dependencies schema, discussed in de Marneffe et al. (2013).

Annotation proceeded in several phases. In the first phase, each annotator made a pass through a separate piece of the

corpus and brought any difficult annotation decisions to the attention of the group to give us an opportunity to identify the challenges of working on web data. After this, we moved to a round of double annotation, in which different pairs of annotators independently annotated a small batch of data each, for a total of 6,670 double-annotated tokens. We measured interannotator agreement at 94%, and subsequently discussed and resolved all of the interannotator disagreements.

After this initial stage of training and establishing consistency, we proceeded to single-annotate most of the data, while still flagging and discussing hard cases and incorporating any resulting decisions into the annotation standard. Decisions about the standard that resulted in broadly applicable changes (such as merges of relations, or the structural change in the treatment of copulas, both of which are discussed in Section 4.) were also implemented retroactively by automatically normalizing all annotated text to any systematic changes in naming and usage of dependency relations.

Currently 227,908 tokens total have been annotated. Between finishing annotation of those tokens and the publication of this paper, interannotator agreement was measured at 96% for 1,025 tokens. From this small sample and from the experience of continually discussing annotations, it is clear to us that relations at the clausal level (specifically, *xcomp vs. acomp*, and *parataxis vs. conj*) seem to be the main source of conflicting annotations.

We do not consider the annotation process finished. In the next phase of this project, we will perform further error analysis on a larger sample of double-annotated text to identify significant sources of disagreement. That information will then guide systematic revisions of particular relations across the corpus, in order to ensure consistency.

4. Changes to the SD standard

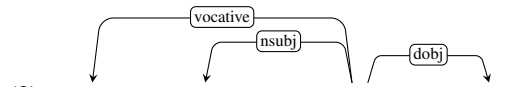
The SD standard was originally designed for use in converting from constituency trees into dependency trees, and this work represents the first large-scale manual annotation project to apply the standard directly. This new perspective led to some revisions of the inventory of dependency types. Some of these revisions are discussed in de Marneffe et al. (2013).

Because the target genres are quite different than the genres that the schema was originally developed for, in the course of the annotation process, some relations between words did not seem to fit into any of the existing dependency types. At the same time, the extensive manual annotation made it clear that some distinctions made by the current standard are not practical. This section highlights some of the changes to the set of dependency labels that were proposed in the course of the EWT annotation.

The SD standard, which is English-centric, is also under revision for cross-linguistic adequacy, as described in (de Marneffe et al., 2014). The work on a Universal Stanford Dependencies (USD) standard is currently in progress and is only partly reflected at the moment in the annotation of the EWT. However, we do plan to work on making the EWT gold standard consistent with the USD standard in the future by implementing the changes that affect English annotations—notably, the new treatment of prepositional phrases, which have lexical heads in their USD representation.

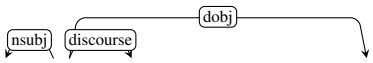
4.1. New relations

In typical newswire text, there are hardly any instances of vocatives—constituents that identify the addressee. However, this occurs frequently enough in the less formal contexts often found on the web to warrant its own dependency label, which we have decided to call *vocative*, illustrated in 2, from the weblog subcorpus.



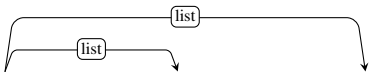
(2) Tronicus , if you 're still checking this thread

Similarly, there are some discourse elements that occur frequently in web data but rarely in journal articles, such as *'uh'* and *'um'*, *'yeah'*, *'hey'*, *'please'*. For those, we introduce the label *discourse*. 3 shows an example, also from weblog; note the lack of capitalization of *i*, typical of this genre.



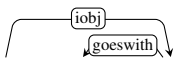
(3) i count like 14 explosive headlines .

We also see contact lists in emails and forum messages; for those, we introduce *list*, which is used to internally structure contact information. That relation is exemplified in (4):



(4) Bill Schmidt – animals@aol.com – 732-657-3416

Finally, typos are far more frequent in web text than in journals. One form of typo that is particularly troublesome for this type of annotation is when a word is split up with a space, creating two tokens. Both tokens need to be annotated with a dependency relation. The POS tag GW was introduced in the tree annotation of the EWT for this phenomenon specifically; in response to that, we included a *goeswith* relation, to indicate that a word fragment goes with the preceding word, demonstrated in 5, from reviews.



(5) Do your self a favor

4.2. Removed relations

The standard was also changed in the direction of removing relations, consistent with some of the changes being made for USD. In particular, *attr* is no longer a part of the SD formalism. The use of *attr* represented a minor inconsistency in the treatment of copular constructions, and the analyses that involved that relation were restructured in terms of *cop*.

4.3. Merged relations

Some distinctions between relation types were removed from the standard, either because they seemed unnecessary or because they were too difficult to make automatically. For example, purpose clauses were marked with type

purpcl. However, this is a special distinction for a particular subtype of *advcl*; in addition to being hard to justify, the distinction is impractical to learn automatically. For those reasons, *purpcl* was subsumed by *advcl*. We also propose that *mark* subsume *complm* and that *appos* subsume *abbrev*. The distinction between *partmod* or *infmod* was dropped in favor of the new type *vmod*, and the new type *discourse* subsumes *intj*.

4.3.1. Advanced Constructions

In addition to defining new relations, we sometimes had to create sensible dependency structures for difficult syntactic constructions, including tough-movement, correlative constructions, and comparatives. These analyses and the arguments that support them are discussed in more detail in (de Marneffe et al., 2013).

5. Changes to the converter

The development of the gold standard for the EWT corpus was used to inform the development of the Stanford converter. The converter works by finding the head of a tree, finding the head of its subtrees, and connecting them by matching patterns with a tool called Tregex (Levy and Andrew, 2006), an analog to regular expressions that applies to trees. This means that dependency typing is based on constituent configurations: we take specific patterns in the constituent trees to represent realizations of certain syntactic dependencies. Extensive expert work has gone into hand-building patterns for dependency typing.

5.1. Error reduction

In some cases, the annotation process revealed systematic inaccuracies in the converter output, pointing back to problems with the Tregex patterns. We are currently implementing changes to the converter that address those systematicities; in this subsection, we discuss some of the changes already in place.

In order to reduce the number of errors that had to be corrected by the annotators, these improvements were implemented during the annotation process, and over time annotators worked on the output of different versions of the converter. As a result, the frequency of each error in the whole corpus is not available; therefore, we are able to report error reduction only relative to the errors that were identified by annotators, in the section of the corpus that was annotated before the correction was implemented.

5.1.1. Addition of rules for constructions

One of the main issues in the functioning of the Stanford converter is the failure to type a dependency, which results in the underspecified label *dep*. We were able to obtain improved performance by addressing some common configurations where *dep* was assigned. Some examples of constructions where formerly *dep* was assigned and which are now correctly types are: determiners attached to adjectives (such as in *'the rich'*, where now we have *det(rich, the)*), emphatic reflexives (such as *'Bush himself'*, now analyzed as *npadvmod(Bush, himself)*) and exclamative inverted clauses (such as *'Such a good idea that was!'*).

Additionally, as mentioned in Section 2. fragments are much more common in this corpus than in the WSJ, which drove the initial stage of development of the converter. As a result, the work with the EWT revealed that in many cases conversion rules relied on the presence of a clausal node and did not work under fragment nodes. To the extent possible, such rules were rewritten to be more general, eliminating a common source of *dep*.

The 2012 version of the converter produced 14803 *dep* relations on the parser training section of PTB (sections 02-21). The current version produces 11685, a reduction of 21%, which shows the better coverage of our extended and improved dependency typing rules.

5.1.2. Improvement in negation attachment

Another issue with correctness in the older versions of the converter was negation attachment. In this example, the converter would incorrectly attach the negation:

(6)

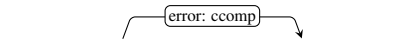
As described above, the converter operates by taking the head of a tree and attaching the heads of the subtrees to it. As the noun phrase *'coal not barbecue charcoal'* is usually parsed as a flat *'NP'*, that would result in *'not'* attaching to *'coal'*.

To fix this, we altered the structure of the tree as a preprocessing step to put *'not barbecue charcoal'* in its own subtree when it occurred after a conjunction, such as in this example. That allows other patterns to work correctly.

5.1.3. Improvement in typing of clausal dependencies

Some constituency configurations were found to be systematically mistyped by the converter, and we have taken steps to correct the more common cases of that.

The *parataxis* dependency type, a loose joining construction for independent clauses that do not have a clear relation to each other, is used very often in the EWT, and was revealed to be another source of error. Annotators noticed that it was often misidentified as *ccomp*, the type of finite complements. Example 7a illustrates the problem.



(7) a.

We refined the conversion rules to identify these structures as paratactic. The primary cause of this was that the search patterns matched both *ccomp* and *parataxis*, with preference given to *ccomp*, so removing the conflicts fixed the incorrect dependencies. Out of 342 errors of this type identified by annotators, only 40 persisted after the converter was changed.

Another relation that posed a challenge for the automatic converter was *vmod*. In the case of purpose clauses headed by *'to'* (which are functionally adverbial, and should be annotated as *vmod* because they lack overt subjects and a finite verb), the converter systematically typed the dependency as *xcomp*, which represents nonfinite complements, as illustrated in (8a).

(8) a.

	Version 1.6.1	Current
Free relative	64.3	64.3
Object extraction from a relative clause	22.0	**34.0**
Object extraction from a reduced relative clause	**1.1**	0.0
Subject extraction from a relative clause	74.7	74.7
Subject extraction from an embedded clause	10.6	**18.0**
Object *wh*-question	41.2	**88.0**
Right node raising	**45.4**	38.0
Average	37.0	**45.3**

Table 2: Converter performance on the test set of the unbounded dependency corpus.

A clause such as this is now correctly identified as *vmod*. The constituency trees do not always contain enough information to reliably distinguish between *vmod* and *xcomp*, however, and so in ambiguous cases the converter uses information about the verb to choose between the relations. Verbs known to usually take nonfinite complements, such as *'ask'* or *'tell'*, trigger the *xcomp* typing. With other verbs, the converter defaults to *vmod*. We plan to extend the list of verbs currently recognized by identifying *xcomp*-taking verbs in the finished gold annotations. This strategy has its limitations, since the list can never be exhaustive, and the same verb used can lead to different analyses in different contexts. However, implementing this change fixed 35% of the errors we found between *vmod* and *xcomp*.

We also found problems in the identification of the *xcomp* relation. In some cases where a verb that takes a direct object and a nonfinite complement, such as *'make'*, is passivized, the converter regularly mistyped the nonfinite complement as *acomp*, the relation used for the object predicates of traditional grammar. This is shown in 9a.

(9) a. Survey responses can not be made (error: acomp) anonymous .

The rule was refined to trigger different dependency types if the main verb is active or passive, and now *'anonymous'* would be correctly identified as *xcomp*. This change resolved 13% of the errors between *acomp* and *xcomp*.

5.2. Conformity with revisions to SD standard

The converter currently implements most but not all changes made to the SD standard (described in Section 4.). All the merges and all removed relations are reflected in the current converter output; however, from the new relations, only *discourse* and *goeswith* are currently produced. We are developing conversion rules for *vocative* and *list*.

5.3. Long-distance dependencies

We also improved the converter on finding long-distance dependencies. Rimell et al. (2009) developed a gold-standard corpus, the unbounded dependency corpus, focusing on 7 long-distance dependency constructions, and evaluated how good different parsers were at retrieving such dependencies. The corpus contains a development set of approximately 20 sentences per construction, and a test set of 80 sentences per construction, in which only the unbounded dependencies have been annotated. Table 2 compares the performance of the Stanford parser and converter (version 1.6.1) reported in Rimell et al. (2009) with its current performance. Overall the results increase, especially in the case of object *wh*-questions. The slight drop in performance in right node raising constructions is due to the fact that we do not spread objects across conjunctions anymore since it created too many false dependencies onto intransitive verbs (e.g., *She ran and waved goodbye* where we do not want to create an object of the verb *ran*). However this results in losing the ability to retrieve objects in right node raising: for example, in *Sue likes but Bill dislikes that TV show*, the converter only retrieves *dobj*(dislikes, show) and omits *dobj*(likes, show). The tensions here illustrate a limitation of Rimell et al.'s metric: It focuses only on recall and does not look for false positives.

Even though this evaluation focuses only on recall, it demonstrates that the converter improved in its ability to recover long-distance dependencies.

5.4. Performance enhancement

We have also been able to make the implementation of the converter faster. The training section of the WSJ could be converted in 1450s on a modern machine using the version from 2012; the current version of the converter takes only 78s. The primary source of the improvement is improvements in the underlying Tregex library, with more compact matching expressions making up a smaller but still substantial fraction of the improvement.

6. Experiments

We demonstrate two uses of this resource: (a) as a gold standard for automatically evaluating the performance of dependency parsers; (b) as a resource for training parsers that will be used on web text. We report experimental results for these two applications.

6.1. Use in automatic dependency conversion evaluation

This set of experiments evaluates the dependency conversion tool included with the Stanford Parser against our gold standard. We use the Stanford Parser to convert the set of EWT trees into dependency representations, and evaluate them against the manually produced dependencies with the official CoNLL Shared Task evaluation script, which reports labeled and unlabeled attachment scores, and label accuracy. Although widely used in NLP, this converter had not been evaluated against a gold standard before.

Training set	No EWT data	Weight=1	Weight=5	Weight=10
EWT Labeled attachment score	80.08	82.80	82.93	82.54
EWT Unlabeled attachment score	83.30	85.46	85.57	85.23
EWT Label accuracy	87.65	89.76	89.78	89.53
WSJ Labeled attachment score	86.85	86.66	86.35	85.87
WSJ Unlabeled attachment score	88.60	88.38	88.12	87.66
WSJ Label accuracy	92.49	92.36	92.08	91.67

Table 3: Parser performance results by training configuration. "Weight=*i*" describes a training set containing *i* copies of the EWT training data, in addition to one copy of the WSJ data.

Training set	Version 2.0	Current
Labeled attachment score	93.36	94.55
Unlabeled attachment score	97.26	97.73
Label accuracy	94.05	95.11

Table 4: Converter performance results.

To assess how the dependency conversion tool provided with the Stanford Parser has evolved throughout the years, we tested two versions: the current internal version, which implements some of the changes motivated by the work on the EWT data; and the 2.0 release. This particular release was chosen because the dependencies produced by it were used as a gold standard for the evaluation other systems in the 2012 First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL) (Petrov and McDonald, 2012).

There are some difficulties in evaluating these versions of the converter, because the gold standard reflects change in the SD standard itself. In terms of evaluating its viability as an annotation tool, it is not helpful to penalize the parser for using labels that were later changed, for example; or, more generally, for failing to produce output that it was not designed to produce. For this reason, we normalized all files to reflect the changes to the standard that could be obtained by automatically rewriting annotations with the previous standard (so, for instance, dependency types subsumed by others were appropriately relabeled).

The accuracy results in Table 4 show that the tool performs well and produces high-quality dependency annotation. The difference in performance between the two versions is small, showing that, in spite of genre differences, the most common structures are present uniformly across different styles of text. We do note, however, that our implementation of changes proposed to the standard is not complete, and we expect a higher performance to follow when the converter incorporates all the insights arising from this annotation effort.

A breakdown of these results by dependency type can be found in Table 5. (Detailed explanations of each dependency type can be found in de Marneffe and Manning (2008).) The table illustrates clearly why increases in performance are not very large when porting results across genres: overwhelmigly common relations such as *det*, *pobj*, or *root* are relations on which both versions do very well. Overall, there is a trend for small improvements in the current version, more often in recall (sometimes at the expense of precision), which reflects the fact that we have worked on extending the set of conversion patterns, and is consistent with the reduction in the frequency of the *dep* relation in the converter output. The only two relations on which the previous version of the converter has better precision and better recall are *advmod* and *expl*, but the differences are very small. The largest improvements relative to version 2.0 are in the relations *parataxis*, which was rarely identified, and *ccomp*, which is used much more precisely now.

6.2. Use in parser training

The purpose of this set of experiments is to demonstrate the importance of training on the target genres for parsing non-canonical language. We trained MaltParser (Nivre, 2003), a system for dependency parsing which can be used to induce a model from data, on two datasets. The first dataset contains sections 2-21 of the OntoNotes 4.0 Release version of the Wall Street Journal (WSJ) (Hovy et al., 2006), converted to SD with the current converter. This version of the WSJ corpus is annotated with the newer set of Penn Treebank tags, making it more closely parallel to the EWT. For the second dataset, we added half of the EWT data (113,985 tokens) to this WSJ data. Because the WSJ corpus is so much larger than the EWT corpus, we experimented with upweighting the EWT data by 5 and 10, creating two more data sets. All the data was normalized. We tested on the remaining half of the EWT data, drawn equally from each subgenre, and on section 22 of the WSJ.

Table 3 shows the results of our preliminary experiments. Adding EWT data improved parser performance on unseen web text, without a major decrease in performance on the WSJ corpus. In spite of the difference in size beween the two corpora, we did not find much benefit in upweighting the EWT data.

7. Conclusion

We presented our work on creating gold standard Stanford dependency annotations for the EWT corpus, a resource for training and evaluating dependency parsers on non-canonical language. We demonstrated the use of this resource both for assessing the accuracy of the automatic dependency conversion tool included with the Stanford Parser, and for improving the accuracy of MaltParser, a direct dependency parser. Our preliminary results substantiate the claims that the Stanford Parser produces high-quality dependency annotations, and that training across genres improves parser performance.

Label	Count	Version 2.0		Current version	
		% Rec	% Prec	% Rec	% Prec
acomp	227	**82.82**	65.96	72.69	**67.35**
advcl	2527	80.02	**95.24**	**84.37**	94.21
advmod	10292	**95.44**	**96.40**	95.42	96.38
amod	10340	97.65	98.69	**97.96**	**98.86**
appos	1847	28.26	67.70	**37.63**	**73.62**
aux	10844	**98.50**	99.41	99.18	**99.42**
auxpass	1516	99.60	99.08	**99.67**	99.08
cc	7005	95.16	96.64	**97.43**	**97.64**
ccomp	2934	82.99	70.91	**84.19**	**86.55**
conj	8117	85.94	96.39	**88.68**	**97.10**
cop	4154	95.50	98.19	**96.05**	**98.81**
csubj	238	47.06	75.17	47.06	75.17
csubjpass	5	80.00	100.00	80.00	100.00
dep	725	**87.31**	8.41	84.83	**10.96**
det	17300	**99.25**	99.42	98.20	**99.67**
discourse	862	79.35	**94.48**	**90.37**	94.20
dobj	10951	94.39	**96.98**	**94.85**	96.89
expl	601	**66.89**	**97.57**	66.72	97.33
goeswith	249	90.76	69.97	**94.78**	**87.73**
iobj	414	75.85	95.73	75.85	**96.02**
mark	3289	91.00	**98.52**	**96.66**	97.85
mwe	297	57.58	96.61	57.58	96.61
neg	1974	90.98	**98.36**	**94.38**	87.10
nn	11054	97.65	97.75	**97.86**	**98.06**
npadvmod	584	55.82	**92.88**	**65.75**	90.78
nsubj	17943	97.85	95.62	**98.25**	**95.99**
nsubjpass	1315	98.71	97.96	**99.09**	**98.19**
num	2629	92.51	**94.45**	**94.98**	94.01
number	250	54.80	38.92	**68.40**	**63.81**
parataxis	1496	5.82	**84.47**	**44.99**	71.07
pcomp	1179	97.12	**95.10**	**97.54**	95.04
pobj	18499	97.98	99.33	**98.57**	99.33
poss	3896	99.36	99.56	**99.49**	**99.67**
possessive	746	99.73	99.73	**100.00**	99.73
preconj	107	82.24	97.78	82.24	**98.88**
predet	179	93.85	98.25	93.85	98.25
prep	19175	96.77	**99.18**	**98.51**	98.91
prt	829	94.45	**99.75**	**97.83**	99.63
punct	2240	94.33	96.31	**97.01**	**96.84**
quantmod	227	**92.07**	78.57	91.63	**79.09**
rcmod	2069	89.56	**97.78**	**90.43**	96.84
root	14676	96.20	96.20	**96.91**	**96.90**
tmod	1077	89.14	95.24	**89.23**	**95.53**
vmod	1765	72.35	**97.56**	**78.30**	81.44
xcomp	3265	**84.84**	84.61	78.62	**87.73**

Table 5: A breakdown of the performance of two versions of the Stanford converter: the 2.0 release, and the current (development) version of the converter by dependency type. *Label* indicates the dependency type; *Count* shows the absolute frequency in the manually annotated portion of the EWT. For each version, we report recall and precision.

In future work, we plan to consolidate our annotations by taking steps to ensure consistency and implementing the representation proposed in the USD standard. When the gold standard is finished, we plan to distribute the resulting corpus as part of uni-dep-tb, the Universal Dependency Treebank Project (`https://code.google.com/p/uni-dep-tb/`). We also plan to use it to learn properties of verbs that can be used in the converter to disambiguate between, for example, clausal adjuncts and complements, as mentioned in Section 5.1.3.

8. Acknowledgements

We thank Google Inc. for a gift that partially supported the production of this treebank and the studies reported herein. Thanks also to Hanzhi Zhu and Daniel Galbraith for help with the corpus annotation.

9. References

Böhmová, A., Hajič, J., Hajičová, E., and Hladká, B. (2003). The Prague dependency treebank. In *Treebanks*, pages 103–127. Springer.

de Marneffe, M.-C. and Manning, C. D. (2008). Stanford typed dependencies manual. Technical report, Stanford University.

de Marneffe, M.-C., MacCartney, B., and Manning, C. D. (2006). Generating typed dependency parses from phrase structure parses. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 449–454.

de Marneffe, M.-C., Connor, M., Silveira, N., Bowman, S. R., Dozat, T., and Manning, C. D. (2013). More constructions, more genres: Extending Stanford dependencies. In *Proceedings of DepLing 2013*.

de Marneffe, M.-C., Dozat, T., Silveira, N., Haverinen, K., Ginter, F., Nivre, J., and Manning, C. D. (2014). Universal stanford dependencies: A cross-linguistic typology. In *Proceedings of the 9th International Conference on Language Resources and Evaluation*.

Hovy, E., Marcus, M., Palmer, M., Ramshaw, L., and Weischedel, R. (2006). Ontonotes: the 90% solution. In *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, NAACL-Short '06, pages 57–60, Stroudsburg, PA, USA. Association for Computational Linguistics.

Klein, D. and Manning, C. D. (2003). Fast exact inference with a factored model for natural language parsing. In *Advances in Neural Information Processing Systems 15 (NIPS)*, pages 3–10. MIT Press.

Levy, R. and Andrew, G. (2006). Tregex and Tsurgeon: Tools for querying and manipulating tree data structures. In *Proceedings of the 5th International Conference on Language Resources and Evaluation*, pages 2231–2234. http://www-nlp.stanford.edu/software/tregex.shtml.

Nivre, J. (2003). An efficient algorithm for projective dependency parsing. In *Proceedings of the 8th International Workshop on Parsing Technologies (IWPT*, pages 149–160.

Petrov, S. and McDonald, R. (2012). Overview of the 2012 shared task on parsing the web. In *Proceedings of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*.

Pyysalo, S., Ginter, F., Haverinen, K., Heimonen, J., Salakoski, T., and Laippala, V. (2007). On the unification of syntactic annotations under the Stanford dependency scheme: A case study on BioInfer and GENIA. In *Proceedings of BioNLP 2007: Biological, translational, and clinical language processing (ACL07)*.

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