# A Dependency Parser for Tweets

**Lingpeng Kong  Nathan Schneider  Swabha Swayamdipta**
**Archna Bhatia  Chris Dyer  Noah A. Smith**
Language Technologies Institute
Carnegie Mellon University
Pittsburgh, PA 15213, USA
{lingpenk,nschneid,swabha,archna,cdyer,nasmith}@cs.cmu.edu

## Abstract

We describe a new dependency parser for English tweets, TWEEBOPARSER. The parser builds on several contributions: new syntactic annotations for a corpus of tweets (TWEEBANK), with conventions informed by the domain; adaptations to a statistical parsing algorithm; and a new approach to exploiting out-of-domain Penn Treebank data. Our experiments show that the parser achieves over 80% unlabeled attachment accuracy on our new, high-quality test set and measure the benefit of our contributions.

Our dataset and parser can be found at http://www.ark.cs.cmu.edu/TweetNLP.

## 1 Introduction

In contrast to the edited, standardized language of traditional publications such as news reports, social media text closely represents language as it is used by people in their everyday lives. These informal texts, which account for ever larger proportions of written content, are of considerable interest to researchers, with applications such as sentiment analysis (Greene and Resnik, 2009; Kouloumpis et al., 2011). However, their often nonstandard content makes them challenging for traditional NLP tools. Among the tools currently available for tweets are a POS tagger (Gimpel et al., 2011; Owoputi et al., 2013) and a named entity recognizer (Ritter et al., 2011)—but not a parser.

Important steps have been taken. The English Web Treebank (Bies et al., 2012) represents an annotation effort on web text—which likely lies somewhere between newspaper text and social media messages in formality and care of editing—that was sufficient to support a shared task (Petrov and McDonald, 2012). Foster et al. (2011b) annotated a small test set of tweets to evaluate parsers trained on the Penn Treebank (Marcus et al., 1993), augmented using semi-supervision and in-domain data. Others, such as Soni et al. (2014), have used existing Penn Treebank–trained models on tweets.

In this work, we argue that the Penn Treebank approach to annotation—while well-matched to edited genres like newswire—is poorly suited to more informal genres. Our starting point is that rapid, small-scale annotation efforts performed by imperfectly-trained annotators should provide enough evidence to train an effective parser. We see this starting point as a necessity, given observations about the rapidly changing nature of tweets (Eisenstein, 2013), the attested difficulties of domain adaptation for parsing (Dredze et al., 2007), and the expense of creating Penn Treebank–style annotations (Marcus et al., 1993).

This paper presents TWEEBOPARSER, the first syntactic dependency parser designed explicitly for English tweets. We developed this parser following current best practices in empirical NLP: we annotate a corpus (TWEEBANK) and train the parameters of a statistical parsing algorithm. Our research contributions include:

- a survey of key challenges posed by syntactic analysis of tweets (by humans or machines) and decisions motivated by those challenges and by our limited annotation-resource scenario (§2);
- our annotation process and quantitative measures of the quality of the annotations (§3);
- adaptations to a statistical dependency parsing algorithm to make it fully compatible with the above, and also to exploit information from out-of-domain data cheaply and without a strong commitment (§4); and
- an experimental analysis of the parser's unlabeled attachment accuracy—which surpasses 80%—and contributions of various important components (§5).

The dataset and parser can be found at http://www.ark.cs.cmu.edu/TweetNLP.



*Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1001–1012,
October 25-29, 2014, Doha, Qatar. 

## 2 Annotation Challenges

Before describing our annotated corpus of tweets (§3), we illustrate some of the challenges of syntactic analysis they present. These challenges motivate an approach to annotation that diverges significantly from conventional approaches to treebanking. Figure 1 presents a single example illustrating four of these: token selection, multiword expressions, multiple roots, and structure within noun phrases. We discuss each in turn.

### 2.1 Token Selection

Many elements in tweets have no syntactic function. These include, in many cases, hashtags, URLs, and emoticons. For example:

RT @justinbieber : now Hailee get a twitter

The retweet discourse marker, username, and colon should not, we argue, be included in the syntactic analysis. By contrast, consider:

Got #college admissions questions ? Ask them tonight during #CampusChat I'm looking forward to advice from @collegevisit http://bit.ly/cch0Tk

Here, both the hashtags and the at-mentioned username are syntactically part of the utterances, while the punctuation and the hyperlink are not. In the example of Figure 1, the unselected tokens include several punctuation tokens and the final token #believer, which marks the topic of the tweet.

Typically, dependency parsing evaluations ignore punctuation token attachment (Buchholz and Marsi, 2006), and we believe it is a waste of annotator (and parser) time to decide how to attach punctuation and other non-syntactic tokens. Ma et al. (2014) recently proposed to treat punctuation as context features rather than dependents, and found that this led to state-of-the-art performance in a transition-based parser. A small adaptation to our graph-based parsing approach, described in §4.2, allows a similar treatment.

Our approach to annotation (§3) forces annotators to explicitly select tokens that have a syntactic function. 75.6% tokens were selected by the annotators. Against the annotators' gold standard, we found that a simple rule-based filter for usernames, hashtags, punctuation, and retweet tokens achieves 95.2% (with gold-standard POS tags) and 95.1% (with automatic POS tags) average accuracy in the task of selecting tokens with a syntactic function in a ten-fold cross-validation experiment. To take context into account, we developed a first-order sequence model and found that it achieves 97.4% average accuracy (again, ten-fold cross-validated) with either gold-standard or automatic POS tags. Features include POS; shape features that recognize the retweet marker, hashtags, usernames, and hyperlinks; capitalization; and a binary feature for tokens that include punctuation. We trained the model using the structured perceptron (Collins, 2002).

### 2.2 Multiword Expressions

We consider multiword expressions (MWEs) of two kinds. The first, proper names, have been widely modeled for information extraction purposes, and even incorporated into parsing (Finkel and Manning, 2009). (An example found in Figure 1 is LA Times.) The second, lexical idioms, have been a "pain in the neck" for many years (Sag et al., 2002) and have recently received shallow treatment in NLP (Baldwin and Kim, 2010; Constant and Sigogne, 2011; Schneider et al., 2014). Constant et al. (2012), Green et al. (2012), Candito and Constant (2014), and Le Roux et al. (2014) considered MWEs in parsing. Figure 1 provides LA Times and All the Rage as examples.

Penn Treebank–style syntactic analysis (and dependency representations derived from it) does not give first-class treatment to this phenomenon, though there is precedent for marking multiword lexical units and certain kinds of idiomatic relationships (Hajič et al., 2012; Abeillé et al., 2003).[1]

We argue that internal analysis of MWEs is not critical for many downstream applications, and therefore annotators should not expend energy on developing and respecting conventions (or making arbitrary decisions) within syntactically opaque or idiosyncratic units. We therefore allow annotators to decide to group words as explicit MWEs, including: proper names (Justin Bieber, World Series), noncompositional or entrenched nominal compounds (belly button, grilled cheese), connectives (as well as), prepositions (out of), adverbials (so far), and idioms (giving up, make sure).

From an annotator's perspective, a MWE functions as a *single node* in the dependency parse, with no internal structure. For idioms whose internal syntax *is* easily characterized, the parse can be used to capture compositional structure, an at-

[1] The popular Stanford typed dependencies (de Marneffe and Manning, 2008) scheme includes a special dependency type for multiwords, though this is only applied to a small list.

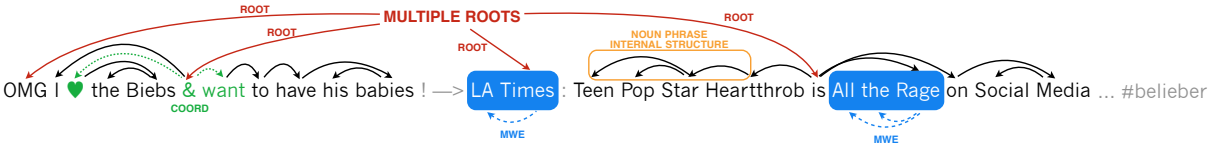



**Figure 1:** Parse tree for a (constructed) example illustrating annotation challenges discussed in §2. Colors highlight token selection (gray; §2.1), multiword expressions (blue; §2.2), multiple roots (red; §2.3), coordination (dotted arcs, green; §3.2), and noun phrase internal structure (orange; §2.4). The internal structure of multiword expressions (dashed arcs below the sentence) was predicted automatically by a parser, as described in §2.2.

tractive property from the perspective of semantic processing.

To allow training a fairly conventional statistical dependency parser from these annotations, we find it expedient to apply an automatic conversion to the MWE annotations, in the spirit of Johnson (1998). We apply an existing dependency parser, the first-order TurboParser (Martins et al., 2009) trained on the Penn Treebank, to parse each MWE independently, assigning structures like those for LA Times and All the Rage in Figure 1. Arcs involving the MWE in the annotation are then reconnected to the MWE-internal root, so that the resulting tree respects the original tokenization. The MWE-internal arcs are given a special label so that the transformation can be reversed and MWEs reconstructed from parser output.

### 2.3 Multiple Roots

For news text such as that found in the Penn Treebank, sentence segmentation is generally considered a very easy task (Reynar and Ratnaparkhi, 1997). Tweets, however, often contain multiple sentences or fragments, which we call "utterances," each with its own syntactic root disconnected from the others. The selected tokens in Figure 1 comprise four utterances.

Our approach to annotation allows multiple utterances to emerge directly from the connectedness properties of the graph implied by an annotator's decisions. Our parser allows multiple attachments to the "wall" symbol, so that multi-rooted analyses can be predicted.

### 2.4 Noun Phrase Internal Structure

A potentially important drawback of deriving dependency structures from phrase-structure annotations, as is typically done using the Penn Treebank, is that flat annotations lead to loss of information. This is especially notable for noun phrases in the Penn Treebank (Vadas and Curran, 2007). Consider Teen Pop Star Heartthrob in Figure 1; Penn Treebank conventions would label this as a single NP with four NN children and no internal structure. Dependency conversion tools would likely attach the first three words in the NP to Heartthrob. Direct dependency annotation (rather than phrase-structure annotation followed by automatic conversion) allows a richer treatment of such structures, which is potentially important for semantic analysis (Vecchi et al., 2013).

## 3 A Twitter Dependency Corpus

In this section, we describe the TWEEBANK corpus, highlighting data selection (§3.1), the annotation process (§3.2), important convention choices (§3.3), and measures of quality (§3.4).

### 3.1 Data Selection

We added manual dependency parses to 929 tweets (12,318 tokens) drawn from the POS-tagged Twitter corpus of Owoputi et al. (2013), which are tokenized and contain manually annotated POS tags.

Owoputi et al.'s data consists of two parts. The first, originally annotated by Gimpel et al. (2011), consists of tweets sampled from a particular day, October 27, 2010—this is known as OCT27. Due to concerns about overfitting to phenomena specific to that day (e.g., tweets about a particular sports game), Owoputi et al. (2013) created a new set of 547 tweets (DAILY547) consisting of one random English tweet per day from January 2011 through June 2012.

Our corpus is drawn roughly equally from OCT27 and DAILY547.[2] Despite its obvious temporal skew, there is no reason to believe this sample is otherwise biased; our experiments in §5 suggest that this property is important.

### 3.2 Annotation

Unlike a typical treebanking project, which may take years and involve thousands of person-hours of work by linguists, most of TWEEBANK was built in a day by two dozen annotators, most of whom had only cursory training in the annotation scheme.

[2]This results from a long-term goal to fully annotate both.

(1) RT @FRIENDSHIP : Friendship is love without kissing ...
```
Friendship > is < love < without < kissing
```

(2) bieber is an alien ! :O he went down to earth .
```
bieber > is** < alien < an
he > [went down]** < to < earth
```

(3) RT @YourFavWhiteGuy : Helppp meeeee . I'mmm meltiiinngggg → http://twitpic.com/316cjg
```
Helppp** < meeeee
I'mmm** < meltiiinngggg
```

**Figure 2:** Examples of GFL annotations from the corpus.

Our annotators used the Graph Fragment Language (GFL), a text-based notation that facilitates keyboard entry of parses (Schneider et al., 2013). A Python Flask web application allows the annotator to validate and visualize each parse (Mordowanec et al., 2014). Some examples are shown in Figure 2. Note that all of the challenges in §2 are handled easily by GFL notation: "retweet" information, punctuation, and a URL are not selected by virtue of their exclusion from the GFL expression; in (2) went down is annotated as a MWE using GFL's square bracket notation; in (3) the tokens are grouped into two utterances whose roots are marked by the ** symbol.

Schneider et al.'s GFL offers some additional features, only some of which we made use of in this project. One important feature allows an annotator to leave the parse *underspecified* in some ways. We allowed our annotators to make use of this feature; however, we excluded from our training and testing data any parse that was incomplete (i.e., any parse that contained multiple disconnected fragments with no explicit root, excluding unselected tokens). Learning to parse from incomplete annotations is a fascinating topic explored in the past (Hwa, 2001; Pereira and Schabes, 1992) and, in the case of tweets, left for future work.

An important feature of GFL that we did use is special notation for coordination structures. For the coordination structure in Figure 1, for example, the notation is:

```
$a :: {♥ want} :: {&}
```

where `$a` creates a new node in the parse tree as it is visualized for the annotator, and this new node attaches to the syntactic parent of the conjoined structure, avoiding the classic forced choice between coordinator and conjunct as parent. For learning to parse, we transform GFL's coordination structures into specially-labeled dependency parses collapsing nodes like `$a` with the coordinator and labeling the attachments specially for postprocessing, following Schneider et al. (2013). In our evaluation (§5), these are treated like other attachments.

### 3.3 Annotation Conventions

A wide range of dependency conventions are in use; in many cases these are *conversion* conventions specifying how dependency trees can be derived from phrase-structure trees. For English, the most popular are due to Yamada and Matsumoto (2003) and de Marneffe and Manning (2008), known as "Yamada-Matsumoto" (YM) and "Stanford" dependencies, respectively. The main differences between them are in whether the auxiliary is the parent of the main verb (or vice versa) and whether the preposition or its argument heads a prepositional phrase (Elming et al., 2013).

A full discussion of our annotation conventions is out of scope. We largely followed the conventions suggested by Schneider et al. (2013), which in turn are close to those of YM. Auxiliary verbs are parents of main verbs, and prepositions are parents of their arguments. The key differences from YM are in coordination structures (discussed in §3.2; YM makes the first conjunct the head) and possessive structures, in which the possessor is the child of the clitic, which is the child of the semantic head, e.g., `the > king > 's > horses`.

### 3.4 Intrinsic Quality

Our approach to developing this initial corpus of syntactically annotated tweets was informed by an aversion to making the perfect the enemy of the good; that is, we sought enough data of sufficient quality to build a usable parser within a relatively short amount of time. If our research goals had been to develop a replicable process for annotation, more training and more quality control would have been called for. Under our budgeted time and annotator resources, this overhead was simply too costly. Nonetheless, we performed a few analyses that give a general picture of the quality of the annotations.

**Inter-annotator agreement.** 170 of the tweets were annotated by multiple users. By the *softComPrec* measure (Schneider et al., 2013),[3] the agreement rate on dependencies is above 90%.

**Expert linguistic judgment.** A linguist coauthor examined a stratified sample (balanced

[3] *softComPrec* is a generalization of attachment accuracy that handles unselected tokens and MWEs.

across annotators) of 60 annotations and rated their quality on a 5-point scale. 30 annotations were deemed to have "no obvious errors," 15 only minor errors, 3 a major error (i.e., clear violation of annotation guidelines),[4] 4 a major error and at least one minor error, and 8 as containing multiple major errors. Thus, 75% are judged as having no major errors. We found this encouraging, considering that this sample is skewed in favor of people who annotated less (including many of the less experienced and/or lower-proficiency annotators).

**Pairwise ranking.** For 170 of the doubly annotated tweets, an experienced annotator examined whether one or the other was markedly better. In 100 cases the two annotations were of comparable quality (neither was obviously better) and did not contain any obvious major errors. In only 7 pairs did both of the annotations contain a serious error.

Qualitatively, we found several unsurprising sources of error or disagreement, including embedded/subordinate clauses, subject-auxiliary inversion, predeterminers, and adverbial modifiers following a modal/auxiliary verb and a main verb. Clarification of the conventions, or even explicit rule-based checking in the validation step, might lead to quality improvements in further annotation efforts.

## 4 Parsing Algorithm

For parsing, we start with TurboParser, which is open-source and has been found to perform well on a range of parsing problems in different languages (Martins et al., 2013; Kong and Smith, 2014). The underlying model allows for flexible incorporation of new features and changes to specification in the output space. We briefly review the key ideas in TurboParser (§4.1), then describe decoder modifications required for our problem (§4.2). We then discuss features we added to TurboParser (§4.3).

### 4.1 TurboParser

Let an input sentence be denoted by $x$ and the set of possible dependency parses for $x$ be denoted by $\mathcal{Y}_x$. A generic linear scoring function based on a feature vector representation $\mathbf{g}$ is used in parsing algorithms that seek to find:

$$\text{parse}^*(x) = \arg\max_{y \in \mathcal{Y}_x} \mathbf{w}^\top \mathbf{g}(x, y) \quad (1)$$

The score is parameterized by a vector $\mathbf{w}$ of weights, which are learned from data (most commonly using MIRA, McDonald et al., 2005a).

The decomposition of the features into local "parts" is a critical choice affecting the computational difficulty of solving Eq. 1. The most aggressive decomposition leads to an "arc-factored" or "first-order" model, which permits exact, efficient solution of Eq. 1 using spanning tree algorithms (McDonald et al., 2005b) or, with a projectivity constraint, dynamic programming (Eisner, 1996).

Second- and third-order models have also been introduced, typically relying on approximations, since less-local features increase the computational cost, sometimes to the point of NP-hardness (McDonald and Satta, 2007). TurboParser attacks the parsing problem using a compact integer linear programming (ILP) representation of Eq. 1 (Martins et al., 2009), then employing alternating directions dual decomposition (AD$^3$; Martins et al., 2011). This enables inclusion of second-order features (e.g., on a word with its sibling or grandparent; Carreras, 2007) and third-order features (e.g., a word with its parent, grandparent, and a sibling, or with its parent and two siblings; Koo and Collins, 2010).

For a collection of (possibly overlapping) parts for input $x$, $\mathcal{S}_x$ (which includes the union of all parts of all trees in $\mathcal{Y}_x$), we will use the following notation. Let

$$\mathbf{g}(x, y) = \sum_{s \in \mathcal{S}_x} \mathbf{f}_s(x, y), \quad (2)$$

where $\mathbf{f}_s$ only considers part $s$ and is nonzero only if $s$ is present in $y$. In the ILP framework, each $s$ has a corresponding binary variable $z_s$ indicating whether part $s$ is included in the output. A collection of constraints relating $z_s$ define the set of feasible vectors $\mathbf{z}$ that correspond to valid outputs and enfore agreement between parts that overlap. Many different versions of these constraints have been studied (Riedel and Clarke, 2006; Smith and Eisner, 2008; Martins et al., 2009, 2010).

A key attraction of TurboParser is that many overlapping parts can be handled, making use of separate combinatorial algorithms for efficiently handling *subsets* of constraints. For example, the constraints that force $\mathbf{z}$ to encode a valid tree can be exploited within the framework by making calls

[4] What we deemed *major* errors included, for example, an incorrect dependency relation between an auxiliary verb and the main verb (like `ima > [have to]`). *Minor* errors included an incorrect attachment between two modifiers of the same head, as in `the > only > [grocery store]`—the correct annotation would have two attachments to a single head, i.e. `the > [grocery store] < only` (or equivalent).

to classic arborescence algorithms (Chu and Liu, 1965; Edmonds, 1967). As a result, when describing modifications to TurboParser, we need only to explain additional constraints and features imposed on *parts*.

### 4.2 Adapted Parse Parts

The first collection of parts we adapt are simple arcs, each consisting of an ordered pair of indices of words in $x$; $\text{arc}(p,c)$ corresponds to the attachment of $x_c$ as a child of $x_p$ (iff $z_{\text{arc}(p,c)} = 1$). Our representation explicitly excludes some tokens from being part of the syntactic analysis (§2.1); to handle this, we constrain $z_{\text{arc}(i,j)} = 0$ whenever $x_i$ or $x_j$ is excluded.

The implication is that excluded tokens are still "visible" to feature functions that involve other edges. For example, some conventional first-order features consider the tokens occurring *between* a parent and child. Even if a token plays no syntactic role of its own, it might still be informative about the syntactic relationships among other tokens. We note three alternative methods:

1. We might remove all unselected tokens from $x$ before running the parser. In §5.6 we find this method to fare 1.7–2.3% worse than our modified decoding algorithm.
2. We might remove unselected tokens but use them to define new features, so that they still serve as evidence. This is the approach taken by Ma et al. (2014) for punctuation. We judge our simple modification to the decoding algorithm to be more expedient, and leave the translation of existing context-word features into that framework for future exploration.
3. We might incorporate the token selection decisions into the parser, performing joint inference for selection and parsing. The AD$^3$ algorithm within TurboParser is well-suited to this kind of extension: $z$-variables for each token's selection could be added, and similar scores to those of our token selection sequence model (§2.1) could be integrated into parsing. Given, however, that the sequence model achieves over 97% accuracy, and that perfect token selection would gain only 0.1–1% in parsing accuracy (reported in §5.5), we leave this option for future work as well.

For first-order models, the above change is all that is necessary. For second- and third-order models, TurboParser makes use of head automata, in particular "grand-sibling head automata" that assign scores to word tuples of $x_g$, its child $x_p$, and two of $x_p$'s adjacent children, $x_c$ and $x'_c$ (Koo et al., 2010). The second-order models in our experiments include parts for $\text{sibling}(p,c,c')$ and $\text{grandparent}(p,c,g)$ and use the grand-sibling head automaton to reason about these together. Automata for an unselected $x_p$ or $x_g$, and transitions that consider unselected tokens as children, are eliminated. In order to allow the scores to depend on unselected tokens between $x_c$ and $x'_c$, we added the binned counts of unselected tokens (mostly punctuation) joint with the word form and POS tag of $x_p$ and the POS tag of $x_c$ and $x'_c$ as features scored in the $\text{sibling}(p,c,c')$ part. The changes discussed above comprise the totality of adaptations we made to the TurboParser algorithm; we refer to them as "parsing adaptations" in the experiments.

### 4.3 Additional Features

**Brown clusters.** Owoputi et al. (2013) found that Brown et al. (1992) clusters served as excellent features in Twitter POS tagging. Others have found them useful in parsing (Koo et al., 2008) and other tasks (Turian et al., 2010). We therefore follow Koo et al. in incorporating Brown clusters as features, making use of the publicly available Twitter clusters from Owoputi et al.[5] We use 4 and 6 bit cluster representations to create features wherever POS tags are used, and full bit strings to create features wherever words were used.

**Penn Treebank features.** A potential danger of our choice to "start from scratch" in developing a dependency parser for Twitter is that the resulting annotation conventions, data, and desired output are very different from dependency parses derived from the Penn Treebank. Indeed, Foster et al. (2011a) took a very different approach, applying Penn Treebank conventions in annotation of a test dataset for evaluation of a parser trained using Penn Treebank trees. In §5.4, we replicate, for dependencies, their finding that a Penn Treebank–trained parser is hard to beat *on their dataset*, which was not designed to be topically representative of English Twitter. When we turn to a more realistic dataset like ours, we find the performance of the Penn Treebank–trained parser to be poor.

Nonetheless, it is hard to ignore such a large amount of high-quality syntactic data. We there-

[5] http://www.ark.cs.cmu.edu/TweetNLP/clusters/50mpaths2

fore opted for a simple, stacking-inspired incorporation of Penn Treebank information into our model.[6] We define a feature on every candidate arc whose value is the (quantized) score of the same arc under a first-order model trained on the Penn Treebank converted using head rules that are as close as possible to our conventions (discussed in more detail in §5.1). This lets a Penn Treebank model literally "weigh in" on the parse for a tweet, and lets the learning algorithm determine how much consideration it deserves.

## 5 Experiments

Our experiments quantify the contributions of various components of our approach.

### 5.1 Setup

We consider two test sets. The first, TEST-NEW, consists of 201 tweets from our corpus annotated by the most experienced of our annotators (one of whom is a co-author of this work). Given very limited data, we believe using the highest quality data for measuring performance, and lower-quality data for training, is a sensibly realistic choice.

Our second test set, TEST-FOSTER, is the dataset annotated by Foster et al. (2011b), which consists of 250 sentences. Recall that their corpus was annotated with phrase structures according to Penn Treebank conventions. Conversion to match our annotation conventions was carried out as follows:

1. We used the PennConverter tool with head rule options selected to approximate our annotation conventions as closely as possible.[7]
2. An experienced annotator manually modified the automatically converted trees by:
   (a) Performing token selection (§2.1) to remove the tokens which have no syntactic function.
   (b) Grouping MWEs (§2.2). Here, most of the MWEs are named entities such as Manchester United.
   (c) Attaching the roots of the utterance in tweets to the "wall" symbol (§2.3).[8]
   (d) Recovering the internal structure of the noun phrases.
   (e) Fixing a difference in conventions with respect to subject-auxiliary inversion.[9]

| | TRAIN | TEST-NEW | TEST-FOSTER |
|---|---|---|---|
| tweets | 717 | 201 | $< 250^{\dagger}$ |
| unique tweets | 569 | 201 | $< 250^{\dagger}$ |
| tokens | 9,310 | 2,839 | 2,841 |
| selected tokens | 7,015 | 2,158 | 2,366 |
| types | 3,566 | 1,461 | 1,230 |
| utterances | 1,473 | 429 | 337 |
| multi-root tweets | 398 | 123 | 60 |
| MWEs | 387 | 78 | 109 |

**Table 1:** Statistics of our datasets. (A tweet with $k$ annotations in the training set is counted $k$ times for the totals of tokens, utterances, etc.). $^{\dagger}$TEST-FOSTER contains 250 manually split sentences. The number of tweets should be smaller but is not recoverable from the data release.

We consider two training sets. TRAIN-NEW consists of the remaining 717 tweets from our corpus (§3) annotated by the rest of the annotators. Some of these tweets have annotations from multiple annotators; 11 annotations for tweets that also occurred in TEST-NEW were excluded. TRAIN-PTB is the conventional training set from the Penn Treebank (§2–21). The PennConverter tool was used to extract dependencies, with head rule options selected to approximate our annotation conventions as closely as possible (see footnote 7). The resulting annotations lack the same attention to noun phrase–internal structure (§2.4) and handle subject-auxiliary inversions differently than our data. Part-of-speech tags were coarsened to be compatible with the Twitter POS tags, using the mappings specified by Gimpel et al. (2011).

Statistics for the in-domain datasets are given in Table 1. As we can see in the table, more than half of the tweets in our corpus have multiple utterances. The out-of-vocabulary rate for our TRAIN/TEST-NEW split is 33.7% by token and 62.5% by type; for TRAIN/TEST-FOSTER it is 41.4% and 64.6% respectively. These are much higher than the 2.5% and 13.2% in the standard Penn Treebank split.

All evaluations here are on unlabeled attachment $F_1$ scores.[10] Our parser provides labels for coordination structures and MWEs (§2), but we do not present detailed evaluations of those due to space constraints.

[6] Stacking is a machine learning method where the predictions of one model are used to create features for another. The second model may be from a different family. Stacking has been found successful for dependency parsing by Nivre and McDonald (2008) and Martins et al. (2008). Johansson (2013) describes further advances that use path features.

[7] `http://nlp.cs.lth.se/software/treebank_converter`; run with `-rightBranching=false -coordStructure=prague -prepAsHead=true -posAsHead=true -subAsHead=true -imAsHead=true -whAsHead=false`.

[8] This was infrequent; their annotations split most multi-utterance tweets into separate sentence-instances.

[9] For example, in the sentence Is he driving, we attached he to driving while PennConverter attaches it to Is.

[10] Because of token selection, precision and recall may not be equal.

### 5.2 Preprocessing

Because some of the tweets in our test set were also in the training set of Owoputi et al. (2013), we retrained their POS tagger on all the annotated data they have minus the 201 tweets in our test set. Its tagging accuracy was 92.8% and 88.7% on TEST-NEW and TEST-FOSTER, respectively. The token selection model (§2.1) achieves 97.4% on TEST-NEW with gold or automatic POS tagging; and on TEST-FOSTER, 99.0% and 99.5% with gold and automatic POS tagging, respectively.

As noted in §4.3, Penn Treebank features were developed using a first-order TurboParser trained on TRAIN-PTB; Brown clusters were included in computing these Penn Treebank features if they were available in the parser to which the features (i.e. Brown clusters) were added.

### 5.3 Main Parser

The second-order TurboParser described in §4, trained on TRAIN-NEW (default hyperparameter values), achieves 80.9% unlabeled attachment accuracy on TEST-NEW and 76.1% on TEST-FOSTER. The experiments consider variations on this main approach, which is the version released as TWEEBOPARSER.

The discrepancy between the two test sets is easily explained: as noted in §3.1, the dataset from which our tweets are drawn was designed to be representative of English on Twitter. Foster et al. (2011b) selected tweets from Bermingham and Smeaton's (2010) corpus, which uses fifty predefined topics like politics, business, sports, and entertainment—in short, topics not unlike those found in the Penn Treebank. Relative to the Penn Treebank training set, the by-type out-of-vocabulary rates are 45.2% for TEST-NEW and only 21.6% for TEST-FOSTER (cf. 13.2% for the Penn Treebank test set).

Another mismatch is in the handling of utterances. In our corpus, utterance segmentation emerges from multi-rooted annotations (§2.3). Foster et al. (2011b) manually split each tweet into utterances and treat those as separate instances in their corpus, so that our model trained on often multi-rooted tweets from TRAIN is being tested only on single-rooted utterances.

### 5.4 Experiment: Which Training Set?

We consider the direct use of TRAIN-PTB instead of TRAIN-NEW. Table 2 reports the results on both test sets, with various options. "Baseline" is off-the-shelf second-order TurboParser. We consider augmenting it with Brown cluster features (§4.3; "+ Brown") and then also with the parsing adaptations of §4.2 ("+ Brown & PA"). Another choice is whether to modify the POS tags at test time; the modified version ("mod. POS") maps at-mentions to pronoun, and hashtags and URLs to noun.

| | Unlabeled Attachment $F_1$ (%) | |
|---|---|---|
| | mod. POS | POS as-is |
| **TEST-NEW** | | |
| Baseline | 73.0 | 73.5 |
| + Brown | 73.7 | 73.3 |
| + Brown & PA | 72.9 | 73.1 |
| **TEST-FOSTER** | | |
| Baseline | 76.3 | 75.2 |
| + Brown | 75.5 | 76.7 |
| + Brown & PA | 76.9 | 77.0 |

**Table 2:** Performance of second-order TurboParser trained on TRAIN-PTB, with various preprocessing options. The main parser (§5.3) achieves 80.9% and 76.1% on the two test sets, respectively; see §5.4 for discussion.

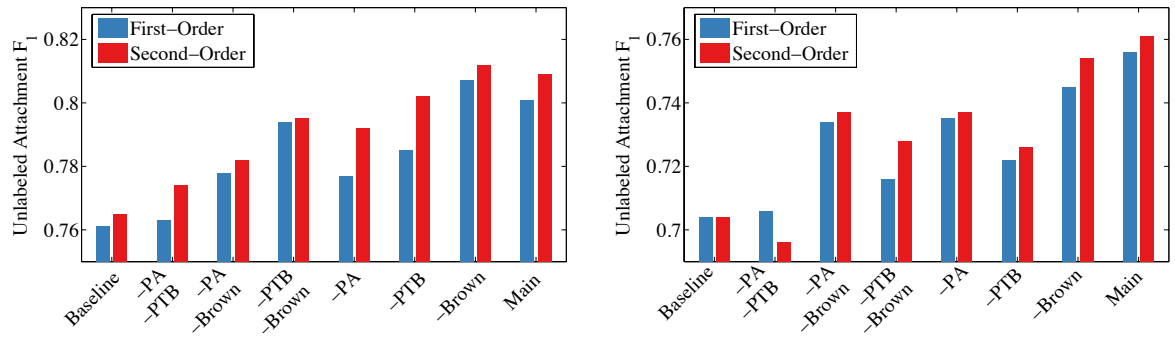
We note that comparing these scores to our main parser (§5.3) conflates three very important independent variables: the amount of training data (39,832 Penn Treebank sentences vs. 1,473 Twitter utterances), the annotation method, and the source of the data. However, we are encouraged that, on what we believe is the superior test set (TEST-NEW), our overall approach obtains a 7.8% gain with an order of magnitude less annotated data.

### 5.5 Experiment: Effect of Preprocessing

Table 3 (second block, italicized) shows the performance of the main parser on both test sets with gold-standard and automatic POS tagging and token selection. On TEST-NEW, with either gold-standard POS tags or gold-standard token selection, performance increases by 1.1%; with both, it increases by 2.3%. On TEST-FOSTER, token selection matters much less, but POS tagging accounts for a drop of more than 6%. This is consistent with Foster et al.'s finding: using a fine-grained Penn Treebank–trained POS tagger (achieving around 84% accuracy on Twitter), they saw 5–8% improvement in unlabeled dependency attachment accuracy using gold-standard POS tags.

### 5.6 Experiment: Ablations

We ablated each key element of our main parser—PTB features, Brown features, second order features and decoding, and the parsing adaptations of




(a) TEST-NEW (b) TEST-FOSTER

**Figure 3:** Feature ablations; these charts present the same scores shown in Table 3 and more variants of the first-order model.

| | Unlabeled Attachment $F_1$ (%) | |
|---|---|---|
| | **TEST-NEW** | **TEST-FOSTER** |
| Main parser | 80.9 | **76.1** |
| *Gold POS and TS* | *83.2* | *82.8* |
| *Gold POS, automatic TS* | *82.0* | *82.3* |
| *Automatic POS, gold TS* | *82.0* | *76.2* |
| Single ablations: | | |
| – PTB | 80.2 | 72.6 |
| – Brown | **81.2** | 75.4 |
| – 2nd order | 80.1 | 75.6 |
| – PA | 79.2 | 73.7 |
| Double ablations: | | |
| – PTB, – Brown | 79.5 | 72.8 |
| – PTB, – 2nd order | 78.5 | 72.2 |
| – PTB, – PA | 77.4 | 69.6 |
| – Brown, – 2nd order | 80.7 | 74.5 |
| – Brown, – PA | 78.2 | 73.7 |
| – 2nd order, – PA | 77.7 | 73.5 |
| Baselines: | | |
| Second order | 76.5 | 70.4 |
| First order | 76.1 | 70.4 |

**Table 3:** Effects of gold-standard POS tagging and token selection (TS; §5.5) and of feature ablation (§5.6). The "baselines" are TurboParser without the parsing adaptations in §4.2 and without Penn Treebank or Brown features. The best result in each column is bolded. See also Figure 3.

§4.2—as well as each pair of these. These conditions use automatic POS tags and token selection. The "– PA" condition, which ablates parsing adaptations, is accomplished by deleting punctuation (in training and test data) and parsing using TurboParser's existing algorithm.

Results are shown in Table 3. Further results with first- and second-order TurboParsers are plotted in Figure 3. Notably, a 2–3% gain is obtained by modifying the parsing algorithm, and our stacking-inspired use of Penn Treebank data contributes in both cases, quite a lot on TEST-FOSTER (unsurprisingly given that test set's similarity to the Penn Treebank). More surprisingly, we find that Brown cluster features do not consistently improve performance, at least not as instantiated here, with our small training set.

## 6 Conclusion

We described TWEEBOPARSER, a dependency parser for English tweets that achieves over 80% unlabeled attachment score on a new, high-quality test set. This is on par with state-of-the-art reported results for news text in Turkish (77.6%; Koo et al., 2010) and Arabic (81.1%; Martins et al., 2011). Our contributions include important steps taken to build the parser: a consideration of the challenges of parsing tweets that informed our annotation process, the resulting new TWEEBANK corpus, adaptations to a statistical parsing algorithm, a new approach to exploiting data in a better-resourced domain (the Penn Treebank), and experimental analysis of the decisions we made. The dataset and parser can be found at `http://www.ark.cs.cmu.edu/TweetNLP`.

## Acknowledgments

The authors thank the anonymous reviewers and André Martins, Yanchuan Sim, Wang Ling, Michael Mordowanec, and Alexander Rush for helpful feedback, as well as the annotators Waleed Ammar, Jason Baldridge, David Bamman, Dallas Card, Shay Cohen, Jesse Dodge, Jeffrey Flanigan, Dan Garrette, Lori Levin, Wang Ling, Bill McDowell, Michael Mordowanec, Brendan O'Connor, Rohan Ramanath, Yanchuan Sim, Liang Sun, Sam Thomson, and Dani Yogatama. This research was supported in part by the U. S. Army Research Laboratory and the U. S. Army Research Office under contract/grant number W911NF-10-1-0533 and by NSF grants IIS-1054319 and IIS-1352440.

## References


Anne Abeillé, Lionel Clément, and François Toussenel. 2003. Building a treebank for French. In *Treebanks*, pages 165–187. Springer.

Timonthy Baldwin and Su Nam Kim. 2010. Multiword expressions. In *Handbook of Natural Language Processing, Second Edition*. CRC Press, Taylor and Francis Group.

Adam Bermingham and Alan F. Smeaton. 2010. Classifying sentiment in microblogs: Is brevity an advantage? In *Proc. of CIKM*.

Ann Bies, Justin Mott, Colin Warner, and Seth Kulick. 2012. English Web Treebank. Technical Report LDC2012T13, Linguistic Data Consortium. URL `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2012T13`.

Peter F Brown, Peter V Desouza, Robert L Mercer, Vincent J. Della Pietra, and Jenifer C. Lai. 1992. Class-based n-gram models of natural language. *Computational Linguistics*, 18(4):467–479.

Sabine Buchholz and Erwin Marsi. 2006. CoNLL-X shared task on multilingual dependency parsing. In *Proc. of CoNLL*.

Marie Candito and Matthieu Constant. 2014. Strategies for contiguous multiword expression analysis and dependency parsing. In *Proc. of ACL*.

Xavier Carreras. 2007. Experiments with a higher-order projective dependency parser. In *Proc. of EMNLP-CoNLL*.

Yoeng-Jin Chu and Tseng-Hong Liu. 1965. On shortest arborescence of a directed graph. *Scientia Sinica*, 14(10):1396.

Michael Collins. 2002. Discriminative training methods for Hidden Markov Models: theory and experiments with perceptron algorithms. In *Proc. of EMNLP*.

Matthieu Constant and Anthony Sigogne. 2011. MWU-aware part-of-speech tagging with a CRF model and lexical resources. In *Proc. of the Workshop on Multiword Expressions: from Parsing and Generation to the Real World*.

Matthieu Constant, Anthony Sigogne, and Patrick Watrin. 2012. Discriminative strategies to integrate multiword expression recognition and parsing. In *Proc. of ACL*.

Marie-Catherine de Marneffe and Christopher D. Manning. 2008. The Stanford typed dependencies representation. In *Proc. of COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

Mark Dredze, John Blitzer, Partha Pratim Talukdar, Kuzman Ganchev, Joao Graca, and Fernando Pereira. 2007. Frustratingly hard domain adaptation for dependency parsing. In *Proc. of EMNLP-CoNLL*.

Jack Edmonds. 1967. Optimum branchings. *Journal of Research of the National Bureau of Standards B*, 71(233-240):160.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proc. of NAACL-HLT*.

Jason Eisner. 1996. Three new probabilistic models for dependency parsing: An exploration. In *Proc. of COLING*.

Jakob Elming, Anders Johannsen, Sigrid Klerke, Emanuele Lapponi, Héctor Martínez Alonso, and Anders Søgaard. 2013. Down-stream effects of tree-to-dependency conversions. In *Proc. of NAACL-HLT*.

Jenny Rose Finkel and Christopher D. Manning. 2009. Joint parsing and named entity recognition. In *Proc of ACL-HLT*.

Jennifer Foster, Özlem Çetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011a. #hardtoparse: POS tagging and parsing the Twitterverse. In *Proc. of AAAI Workshop on Analyzing Microtext*.

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011b. From news to comment: resources and benchmarks for parsing the language of Web 2.0. In *Proc. of IJCNLP*.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proc. of ACL-HLT*.

Spence Green, Marie-Catherine de Marneffe, and Christopher D. Manning. 2012. Parsing models for identifying multiword expressions. *Computational Linguistics*, 39(1):195–227.

Stephan Greene and Philip Resnik. 2009. Syntactic packaging and implicit sentiment. In *Proc. of NAACL*.

Jan Hajič, Eva Hajičová, Jarmila Panevová, Petr Sgall, Silvie Cinková, Eva Fučíková, Marie Mikulová, Petr Pajas, Jan Popelka, Jiří Semecký, Jana Šindlerová, Jan Štěpánek, Josef Toman, Zdeňka Urešová, and Zdeněk Žabokrtský. 2012. Prague Czech-English Dependency Treebank 2.0. Technical Report LDC2012T08, Linguistic Data Consortium. URL `http://www.ldc.upenn.edu/Catalog/catalogEntry.jsp?catalogId=LDC2012T08`.

Rebecca Hwa. 2001. *Learning Probabilistic Lexicalized Grammars for Natural Language Processing*. Ph.D. thesis, Harvard University.

Richard Johansson. 2013. Training parsers on incompatible treebanks. In *Proc. of NAACL-HLT*.

Mark Johnson. 1998. PCFG models of linguistic tree representations. *Computational Linguistics*, 24(4):613–632.

Lingpeng Kong and Noah A. Smith. 2014. An empirical comparison of parsing methods for Stanford dependencies. ArXiv:1404.4314.

Terry Koo, Xavier Carreras, and Michael Collins. 2008. Simple semi-supervised dependency parsing. In *Proc. of ACL*.

Terry Koo and Michael Collins. 2010. Efficient third-order dependency parsers. In *Proc. of ACL*.

Terry Koo, Alexander M. Rush, Michael Collins, Tommi Jaakkola, and David Sontag. 2010. Dual decomposition for parsing with non-projective head automata. In *Proc. of EMNLP*.

Efthymios Kouloumpis, Theresa Wilson, and Johanna Moore. 2011. Twitter sentiment analysis: The good the bad and the OMG! In *Proc. of ICWSM*.

Joseph Le Roux, Matthieu Constant, and Antoine Rozenknop. 2014. Syntactic parsing and compound recognition via dual decomposition: application to French. In *Proc. of COLING*.

Ji Ma, Yue Zhang, and Jingbo Zhu. 2014. Punctuation processing for projective dependency parsing. In *Proc. of ACL*.

Mitchell P. Marcus, Mary Ann Marcinkiewicz, and Beatrice Santorini. 1993. Building a large annotated corpus of English: The Penn Treebank. *Computational linguistics*, 19(2):313–330.

André F.T. Martins, Miguel Almeida, and Noah A. Smith. 2013. Turning on the turbo: Fast third-order non-projective turbo parsers. In *Proc. of ACL*.

André F.T. Martins, Dipanjan Das, Noah A. Smith, and Eric P. Xing. 2008. Stacking dependency parsers. In *Proc. of EMNLP*.

André F.T. Martins, Noah A. Smith, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2011. Dual decomposition with many overlapping components. In *Proc. of EMNLP*.

André F.T. Martins, Noah A. Smith, and Eric P. Xing. 2009. Concise integer linear programming formulations for dependency parsing. In *Proc. of ACL-IJCNLP*.

André F.T. Martins, Noah A. Smith, Eric P. Xing, Pedro M.Q. Aguiar, and Mário A.T. Figueiredo. 2010. Turbo parsers: Dependency parsing by approximate variational inference. In *Proc. of EMNLP*.

Ryan McDonald, Koby Crammer, and Fernando Pereira. 2005a. Online large-margin training of dependency parsers. In *Proc. of ACL*.

Ryan McDonald, Fernando Pereira, Kiril Ribarov, and Jan Hajič. 2005b. Non-projective dependency parsing using spanning tree algorithms. In *Proc. of HLT-EMNLP*.

Ryan McDonald and Giorgio Satta. 2007. On the complexity of non-projective data-driven dependency parsing. In *Proc. of IWPT*.

Michael T. Mordowanec, Nathan Schneider, Chris. Dyer, and Noah A. Smith. 2014. Simplified dependency annotations with GFL-Web. In *Proc. of ACL, demonstration track*.

Joakim Nivre and Ryan McDonald. 2008. Integrating graph-based and transition-based dependency parsers. In *Proc. of ACL-HLT*.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proc. of NAACL-HLT*.

Fernando Pereira and Yves Schabes. 1992. Inside-outside reestimation from partially bracketed corpora. In *Proc. of ACL*.

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*.

Jeffrey C. Reynar and Adwait Ratnaparkhi. 1997. A maximum entropy approach to identifying sentence boundaries. In *Proc. of ANLP*.

Sebastian Riedel and James Clarke. 2006. Incremental integer linear programming for non-projective dependency parsing. In *Proc. of EMNLP*.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: an experimental study. In *Proc. of EMNLP*.

Ivan A. Sag, Timothy Baldwin, Francis Bond, Ann Copestake, and Dan Flickinger. 2002. Multiword expressions: A pain in the neck for NLP. In *Proc. of CICLing*.

Nathan Schneider, Emily Danchik, Chris Dyer, and Noah A. Smith. 2014. Discriminative lexical semantic segmentation with gaps: Running the MWE gamut. *Transactions of the Association for Computational Linguistics*, 2:193–206.

Nathan Schneider, Brendan O'Connor, Naomi Saphra, David Bamman, Manaal Faruqui, Noah A. Smith, Chris Dyer, and Jason Baldridge. 2013. A framework for (under)specifying dependency syntax without overloading annotators. In *Proc. of the 7th Linguistic Annotation Workshop and Interoperability with Discourse*.

David A. Smith and Jason Eisner. 2008. Dependency parsing by belief propagation. In *Proc. of EMNLP*.

Sandeep Soni, Tanushree Mitra, Eric Gilbert, and Jacob Eisenstein. 2014. Modeling factuality judgments in social media text. In *Proc. of ACL*.

Joseph Turian, Lev Ratinov, and Yoshua Bengio. 2010. Word representations: a simple and general method for semi-supervised learning. In *Proc. of ACL*.

David Vadas and James Curran. 2007. Adding noun phrase structure to the Penn Treebank. In *Proc. of ACL*.

Eva Maria Vecchi, Roberto Zamparelli, and Marco Baroni. 2013. Studying the recursive behaviour of adjectival modification with compositional distributional semantics. In *Proc. of EMNLP*.

Hiroyasu Yamada and Yuji Matsumoto. 2003. Statistical dependency analysis with support vector machines. In *Proc. of IWPT*.