Parsing Tweets into Universal Dependencies

Yijia Liu
Harbin Institute of Technology
yjliu@ir.hit.edu.cn

Yi Zhu
University of Cambridge
yz568@cam.ac.uk

Wanxiang Che **Bing Qin**
Harbin Institute of Technology

Nathan Schneider
Georgetown University

Noah A. Smith
University of Washington

Abstract

We study the problem of analyzing tweets with Universal Dependencies (UD; Nivre et al., 2016). We extend the UD guidelines to cover special constructions in tweets that affect tokenization, part-of-speech tagging, and labeled dependencies. Using the extended guidelines, we create a new tweet treebank for English (TWEEBANK V2) that is four times larger than the (unlabeled) TWEEBANK V1 introduced by Kong et al. (2014). We characterize the disagreements between our annotators and show that it is challenging to deliver consistent annotation due to ambiguity in understanding and explaining tweets. Nonetheless, using the new treebank, we build a pipeline system to parse raw tweets into UD. To overcome annotation noise without sacrificing computational efficiency, we propose a new method to distill an ensemble of 20 transition-based parsers into a single one. Our parser achieves an improvement of 2.2 in LAS over the un-ensembled baseline and outperforms parsers that are state-of-the-art on other treebanks in both accuracy and speed.

1 Introduction

NLP for social media messages is challenging, because domain adaptation is required, and also because creating annotated datasets (e.g., treebanks) for training and evaluation is hard. Pioneering work by Foster et al. (2011) annotated 7,630 tokens' worth of tweets according to the phrase-structure conventions of the Penn Treebank (PTB; Marcus et al., 1993), enabling conversion to Stanford Dependencies. Kong et al. (2014) further studied the challenges in annotating tweets and presented a tweet treebank (TWEEBANK), consisting of 12,149 tokens and largely following conventions suggested by Schneider et al. (2013), fairly close to Yamada and Matsumoto (2003) dependencies (without labels). Both annotation efforts were highly influenced by the PTB, whose guidelines have good grammatical coverage on newswire. However, when it comes to informal, unedited, user-generated text, the guidelines may leave many annotation decisions unspecified.

Universal Dependencies (Nivre et al., 2016, UD) were introduced to enable consistent annotation across different languages. To allow such consistency, UD was designed to be adaptable to different genres (Wang et al., 2017) and languages (Guo et al., 2015; Ammar et al., 2016). We propose that analyzing the syntax of tweets can benefit from such adaptability. In this paper, we introduce a new English tweet treebank of 55,607 tokens that follows the UD guidelines, but also contends with social media-specific challenges that were not covered by UD guidelines.[1] Our annotation includes tokenization, part-of-speech (POS) tags, and (labeled) Universal Dependencies. We characterize the disagreements among our annotators and find that consistent annotation is still challenging to deliver even with the extended guidelines.

Based on these annotations, we nonetheless designed a pipeline to parse raw tweets into Universal Dependencies. Our pipeline includes: a bidirectional LSTM (bi-LSTM) tokenizer, a word cluster–enhanced POS tagger (following Owoputi et al., 2013), and a stack LSTM parser with character-based word representations (Ballesteros et al., 2015), which we refer to as our "baseline"

[1] We developed our treebank independently of a similar effort for Italian tweets (Sanguinetti et al., 2017). See §2.5 for a comparison.

parser. To overcome the noise in our annotated data and achieve better performance without sacrificing computational efficiency, we distill a 20-parser ensemble into a single greedy parser (Hinton et al., 2015). We show further that learning directly from the exploration of the ensemble parser is more beneficial than learning from the gold standard "oracle" transition sequence. Experimental results show that an improvement of more than 2.2 points in LAS over the baseline parser can be achieved with our distillation method. It outperforms other state-of-the-art parsers in both accuracy and speed.

The contributions of this paper include:

- We study the challenges of annotating tweets in UD (§2) and create a new tweet treebank (TWEEBANK V2), which includes tokenization, part-of-speech tagging, and labeled Universal Dependencies. We also characterize the difficulties of creating such annotation.
- We introduce and evaluate a pipeline system to parse the raw tweet text into Universal Dependencies (§3). Experimental results show that it performs better than a pipeline of the state-of-the-art alternatives.
- We propose a new distillation method for training a greedy parser, leading to better performance than existing methods and without efficiency sacrifices.

Our dataset and system are publicly available at `https://github.com/Oneplus/Tweebank` and `https://github.com/Oneplus/twpipe`.

2 Annotation

We first review TWEEBANK V1 of Kong et al. (2014), the previous largest Twitter dependency annotation effort (§2.1). Then we introduce the differences in our tokenization (§2.2) and part-of-speech (§2.3) (re)annotation with O'Connor et al. (2010) and Gimpel et al. (2011), respectively, on which TWEEBANK V1 was built. We describe our effort of adapting the UD conventions to cover tweet-specific constructions (§2.4). Finally, we present our process of creating a new tweet treebank, TWEEBANK V2, and characterize the difficulties in reaching consistent annotations (§2.6).

2.1 Background: TWEEBANK

The annotation effort we describe stands in contrast to the previous work by Kong et al. (2014). Their aim was the rapid development of a dependency parser for tweets, and to that end they contributed a new annotated corpus, TWEEBANK, consisting of 12,149 tokens. Their annotations added unlabeled dependencies to a portion of the data annotated with POS tags by Gimpel et al. (2011) and Owoputi et al. (2013) after rule-based tokenization (O'Connor et al., 2010). Kong et al. also contributed a system for parsing; we defer the discussion of their parser to §3.

Kong et al.'s rapid, small-scale annotation effort was heavily constrained. It was carried out by annotators with only cursory training, no clear annotation guidelines, and no effort to achieve consensus on controversial cases. Annotators were allowed to underspecify their analyses. Most of the work was done in a very short amount of time (a day). Driven both by the style of the text they sought to annotate and by exigency, some of their annotation conventions included:

- Allowing an annotator to exclude tokens from the dependency tree. A clear criterion for exclusion was not given, but many tokens were excluded because they were deemed "non-syntactic."
- Allowing an annotator to merge a multiword expression into a single node in the dependency tree, with no internal structure. Annotators were allowed to take the same step with noun phrases.
- Allowing multiple roots, since a single tweet might contain more than one sentence.

These conventions were justified on the grounds of making the annotation easier for non-experts, but they must be revisited in our effort to apply UD to tweets.

2.2 Tokenization

Our tokenization strategy lies between the strategy of O'Connor et al. (2010) and that of UD. There is a tradeoff between preservation of original tweet content and respecting the UD guidelines.

The regex-based tokenizer of O'Connor et al. (2010)—which was originally designed for an exploratory search interface called TweetMotif, not for NLP—preserves most whitespace-delimited tokens, including hashtags, at-mentions, emoticons, and unicode glyphs. They also treat contractions and acronyms as whole tokens and do not

split them. UD tokenization,[2] in order to better serve dependency annotation, treats each syntactic word as a token. They therefore more aggressively split clitics from contractions (e.g., *gonna* is tokenized as *gon* and *na*; *its* is tokenized as *it* and *s* when *s* is a copula). But acronyms are not touched in the UD tokenization guidelines. Thus, we follow the UD tokenization for contractions and leave acronyms like *idc* ("I don't care") as a single token.

In the different direction of splitting tokens, UD guidelines also suggest to merge *multi-token words* (e.g., *20 000*) into one single token in some special cases. We witnessed a small number of tweets that contain multi-token words (e.g., *Y O*, and *R E T W E E T*) but didn't combine them for simplicity. Such tokens only account for 0.07% and we use the UD *goeswith* relation to resolve these cases in the dependency annotations.

2.3 Part-of-Speech Annotation

Before turning to UD annotations, we (re)annotated the data with POS tags, for consistency with other UD efforts, which adopt the universal POS tagset.[3] In some cases, non-corresponding tag conflicts arose between the UD English Web Treebank treebank conventions (UD_English-EWT; de Marneffe et al., 2014)[4] and the conventions of Gimpel et al. (2011). In these cases, we always conformed to UD, enabling consistency (e.g., when we exploit the existing UD_English-EWT treebank in our parser for tweets, §3). For example, the nominal URL in Figure 2 is tagged as *other* (X) and + is tagged as *symbol* (SYM) rather than *conjunction* (CCONJ).

Tokens that do not have a syntactic function (see Figure 1, discussed at greater length in the next section) were usually annotated as *other* (X), except for emoticons, which are tagged as *symbol* (SYM), following UD_English-EWT.

Tokens that abbreviate multiple words (such as *idc*) are resolved to the POS of the syntactic head of the expression, following UD conventions (in this example, the head *care* is a verb, so *idc* is tagged as a verb). When the token is not phrasal, we use the POS of the left-most sub-phrase. For example, *mfw* ("my face when") is tagged as a noun (for *face*).

Compared to the effort of Gimpel et al. (2011), our approach simplifies some matters. For example, if a token is not considered syntactic by UD conventions, it gets an *other* (X) tag (Gimpel et al. had more extensive conventions). Other phenomena, like abbreviations, are more complicated for us, as discussed above; Gimpel et al. used a single part of speech for such expressions.

Another important difference follows from the difference in tokenization. As discussed in §2.2, UD calls for more aggressive tokenization than that of O'Connor et al. (2010) which opted out of splitting contractions and possessives. As a consequence of adopting O'Connor et al.'s (2010) tokenization, Gimpel et al. introduced new parts of speech for these cases instead.[5] For us, these tokens must be split, but universal parts of speech can be applied.

2.4 Universal Dependencies Applied to Tweets

We adopt UD version 2 guidelines to annotate the syntax of tweets. In applying UD annotation conventions to tweets, the choices of Kong et al. (2014) must be revisited. We consider the key questions that arose in our annotation effort, and how we resolved them.

Acronym abbreviations. We follow Kong et al. (2014) and annotate the syntax of an acronym as a single word without normalization. Their syntactic functions are decided according to their context. Eisenstein (2013) studied the necessity of normalization in social media text and argued that such normalization is problematic. Our solution to the syntax of abbreviations follows the spirit of his argument. Because abbreviations which clearly carry syntactic functions only constitute 0.06% of the tokens in our dataset, we believe that normalization for acronyms is an unnecessarily complicated step.

Non-syntactic tokens. The major characteristic that distinguishes tweets from standard texts is that a large proportion of tokens don't carry any syntactic function. In our annotation, there are five types of non-syntactic tokens commonly seen in tweets: sentiment emoticons, retweet markers and

[2] http://universaldependencies.org/u/overview/tokenization.html

[3] A revised and extended version of Petrov et al. (2012) with 17 tags.

[4] https://github.com/UniversalDependencies/UD_English-EWT

[5] These tags only account for 2.7% of tokens, leading to concerns about data sparseness in tagging and all downstream analyses.

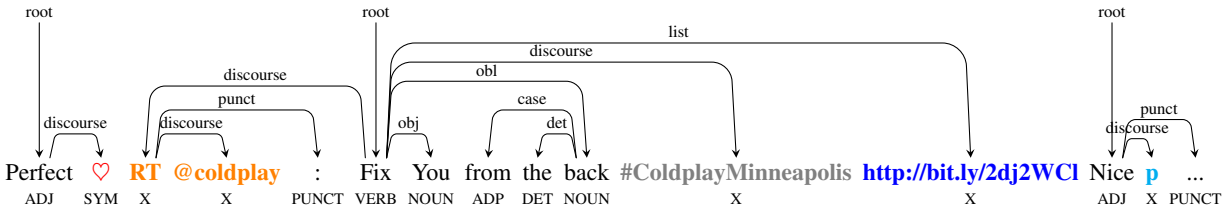

Figure 1: An example to illustrate non-syntactic tokens: sentiment emoticon, retweet marker and its following at-mention, topical hashtag, referential URL, and truncated word. This is a concatenation of three real tweets.

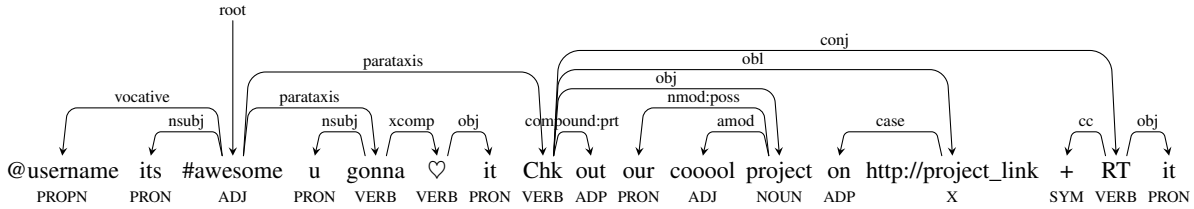

Figure 2: An example to illustrate informal but syntactic tokens. This is a contrived example inspired by several tweets.

	syntactic (%)	non-syntactic (%)
emoticons	0.25	0.95
RT	0.14	2.49
hashtag	1.02	1.24
URL	0.67	2.38
truncated words	0.00	0.49
total	2.08	7.55

Table 1: Proportions of non-syntactic tokens in our annotation. These statistics are obtained on 140 character–limited tweets.

their following at-mentions, topical hashtags, referential URLs, and truncated words.[6] Figure 1 illustrates examples of these non-syntactic tokens. As discussed above, these are generally tagged with the *other* (X) part of speech, except emoticons, which are tagged as *symbol* (SYM). In our annotation, 7.55% of all tokens are belong to one of the five types; detailed statistics can be found in Table 1.

It is important to note that these types may, in some contexts, have syntactic functions. For example, besides being a discourse marker, *RT* can abbreviate the verb *retweet*; emoticons and hashtags may be used as content words within a sentence; and at-mentions can be normal vocative proper nouns: see Figure 2. Therefore, the criteria for annotating a token as non-syntactic must be context-dependent.

Inspired by the way UD deals with *punctuation* (which is canonically non-syntactic), we adopt the following conventions:

- If a non-syntactic token is within a sentence that has a clear predicate, it will be attached to this predicate. The retweet construction is a special case and we will discuss its treatment in the following paragraph.
- If the whole sentence is a sequence of non-syntactic tokens, we attach all these tokens to the first one.
- Non-syntactic tokens are mostly labeled as *discourse*, but URLs are always labeled as *list*, following the UD_English-EWT dataset.

Kong et al. (2014) proposed an additional preprocessing step, *token selection*, in their annotation process. They required the annotators to first select the non-syntactic tokens and exclude them from the final dependency annotation. In order to keep our annotation conventions in line with UD norms and preserve the original tweets as much as possible, we include non-syntactic tokens in our annotation following the conventions above. Compared with Kong et al. (2014), we also gave a clear definition of non-syntactic tokens, which helped us avoid confusion during annotation.

Retweet construction. Figure 1 shows an example of the retweet construction (*RT @coldplay :*). This might be treated as a verb phrase, with *RT* as a verb and the at-mention as an argument. This solution would lead to an uninformative root word

[6]The tweets we analyze have at most 140 characters. Although Twitter has doubled the tweet length limit to 280 characters since our analysis, we believe this type of token will still remain.

and, since this expression is idiomatic to Twitter, might create unnecessary confusion for downstream applications aiming to identify the main predicate(s) of a tweet. We therefore treat the whole expression as non-syntactic, including assigning the *other* (X) part of speech to both *RT* and *@coldplay*, attaching the at-mention to *RT* with the *discourse* label and the colon to *RT* with the *punct*(uation) label, and attaching *RT* to the predicate of the following sentence.

Constructions handled by UD. A number of constructions that are especially common in tweets are handled by UD conventions: ellipsis, irregular word orders, and paratactic phrases and sentences not explicitly delineated by punctuation.

Vocative at-mentions. Another idiomatic construction on Twitter is a vocative at-mention (sometimes a signal that a tweet is a reply to a tweet by the mentioned user). We treat these at-mentions as vocative expressions, labeling them with POS tag *proper noun* (PROPN) and attaching them to the main predicate of the sentence it is within with the label *vocative* as in UD guidelines (see Figure 2 for an example).

2.5 Comparison to PoSTWITA-UD

The first Twitter treebank annotated with Universal Dependencies was the *PosTWITA-UD* corpus for Italian (Sanguinetti et al., 2017), which consists of 6,738 tweets (119,726 tokens). In their convention, tokenization tends to preserve the original tweet content but two special cases, *articulated prepositions* (e.g. *nella* as *in la*) and *clitic clusters* (e.g. *guardandosi* as *guardando si*), are tokenized. Their lemmas include spelling normalization, whereas our lemmas only normalize casing and inflectional morphology. The current UD guidelines on lemmas are flexible, so variation between treebanks is expected.[7]

With respect to tweet-specific constructions, Sanguinetti et al.'s (2017) and our interpretation of headedness is the same, but we differ in the relation label. For topical hashtags, we use *discourse* while Sanguinetti et al. (2017) used *parataxis*. In referential URLs, we use *list* (following the precedent of UD_English-EWT) while Sanguinetti et al. (2017) used *dep*. Our choice of *discourse* for sentiment emoticons is inspired by the observation that emoticons are annotated as *discourse* by UD_English-EWT and Sanguinetti et al. (2017) used the same relation for the emoticons. Retweet constructions and truncated words were not explicitly touched in Sanguinetti et al. (2017). Judging from the released treebank[8], the *RT* marker, at-mention, and colon in the retweet construction are all attached to the predicate of the following sentence with *dep*, *vocative:mention* and *punct*. We expect that the official UD guidelines will eventually adopt standards for these constructions so the treebanks can be harmonized.

[7] http://universaldependencies.org/u/overview/morphology.html#lemmas

	TWEEBANK V1		TWEEBANK V2	
split	tweets	tokens	tweets	tokens
train	639	9,310	1,639	24,753
dev	–	–	710	11,742
test	201	2,839	1,201	19,112
total	840	12,149	3,550	55,607

Table 2: Statistics of TWEEBANK V2 and the comparison with TWEEBANK V1 in quantity.

2.6 TWEEBANK V2

Following the guidelines presented above, we create a new Twitter dependency treebank, which we call TWEEBANK V2.

2.6.1 Data Collection

TWEEBANK V2 is built on the original data of TWEEBANK V1 (840 unique tweets, 639/201 for training/test), along with an additional 210 tweets sampled from the POS-tagged dataset of Gimpel et al. (2011) and 2,500 tweets sampled from the Twitter stream from February 2016 to July 2016.[9] The latter data source consists of 147.4M English tweets after being filtered by the *lang* attribute in the tweet JSON and *langid.py*.[10] In the same way as Kong et al. (2014), reference unit is always the tweet in its entirety – which may thus consist of multiple sentences – not the sentence alone. Before annotation, we use a simple regular expression to anonymize usernames and URLs.

2.6.2 Annotation Process

Our annotation process was conducted in two stages. In the first stage, 18 researchers worked on the TWEEBANK V1 portion and the additional 210 tweets and created the initial annotations in

[8] https://github.com/UniversalDependencies/UD_Italian-PoSTWITA

[9] Data downloaded from https://archive.org/.

[10] https://github.com/saffsd/langid.py

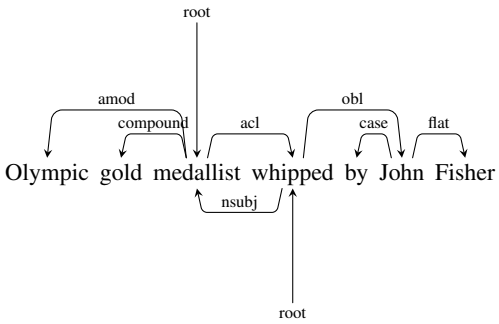

Figure 3: An example of disagreement; one annotator's parse is shown above, disagreeing arcs from the other annotator are shown below. This is a real example in our annotation.

one day. Before annotating, they were given a tutorial overview of the general UD annotation conventions and our guidelines specifically for annotating tweets. Both the guidelines and annotations were further refined by the authors of this paper to increase the coverage of our guidelines and solve inconsistencies between different annotators during this exercise. In the second stage, a tokenizer, a POS tagger, and a parser were trained on the annotated data from the first stage (1,050 tweets in total), and used to automatically analyze the sampled 2,500 tweets. Authors of this paper manually corrected the parsed data and finally achieved 3,550 labeled tweets.[11] Newly created annotations are split into train, development, and test sets and appended to the original splits of TWEEBANK V1. Statistics of our annotations and data splits are shown in Table 2.

We report the inter-annotator agreement between the annotators in the second stage. There is very little disagreement on the tokenization annotation. The agreement rate is 96.6% on POS, 88.8% on unlabeled dependencies, and 84.3% on labeled dependencies. Further analysis shows the major disagreements on POS involve entity names (30.6%) and topical hashtags (18.1%). Taking the example in Figure 1, "Fix you" can be understood as a verbal phrase but also as the name of the Coldplay's single and tagged as proper noun. An example of a disagreement on dependencies is shown in Figure 3. Depending on whether this is an example of a zero copula construction, or a clause-modified noun, either annotation is plausible.

[11] Manual annotation was done with Arborator (Gerdes, 2013), a web platform for drawing dependency trees.

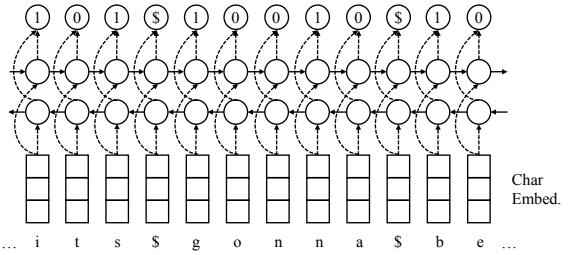

Figure 4: The bi-LSTM tokenizer that segments '*its gonna be*' into '*it s gon na be*'.

System	F_1
Stanford CoreNLP	97.3
Twokenizer	94.6
UDPipe v1.2	97.4
our bi-LSTM tokenizer	98.3

Table 3: Tokenizer comparison on the TWEEBANK V2 test set.

3 Parsing Pipeline

We present a pipeline system to parse tweets into Universal Dependencies. We evaluate each component individually, and the system as a whole.

3.1 Tokenizer

Tokenization, as the initial step of many NLP tasks, is non-trivial for informal tweets, which include hashtags, at-mentions, and emoticons (O'Connor et al., 2010). Context is often required for tokenization decisions; for example, the asterisk in *4*3* is a separate token signifying multiplication, but the asterisk in *sh*t* works as a mask to evoke censorship and should not be segmented.

We introduce a new character-level bidirectional LSTM (bi-LSTM) sequence-labeling model (Huang et al., 2015; Ma and Hovy, 2016) for tokenization. Our model takes the raw sentence and tags each character in this sentence as whether it is the beginning of a word (1 as the beginning and 0 otherwise). Figure 4 shows the architecture of our tokenization model. Space is treated as an input but deterministically assigned a special tag $.

Experimental results. Our preliminary results showed that our model trained on the combination of UD_English-EWT and TWEEBANK V2 outperformed the one trained only on the UD_English-EWT or TWEEBANK V2, consistent with previous work on dialect treebank parsing (Wang et al., 2017). So we trained our tokenizer on the training portion of TWEEBANK V2 combined with the UD_English-EWT training set and tested on the

TWEEBANK V2 test set. We report F_1 scores, combining precision and recall for token identification. Table 3 shows the tokenization results, compared to other available tokenizers. Stanford CoreNLP (Manning et al., 2014) and Twokenizer (O'Connor et al., 2010)[12] are rule-based systems and were not adapted to the UD tokenization scheme. The UDPipe v1.2 (Straka and Straková, 2017) model was re-trained on the same data as our system. Compared with UDPipe, we use an LSTM instead of a GRU in our model and we also use a larger size for hidden units (64 vs. 20), which has stronger representational power. Our bi-LSTM tokenizer achieves the best accuracy among all these tokenizers. These results speak to the value of statistical modeling in tokenization for informal texts.

3.2 Part-of-Speech Tagger

Part-of-speech tagging for tweets has been extensively studied (Ritter et al., 2011; Gimpel et al., 2011; Derczynski et al., 2013; Owoputi et al., 2013; Gui et al., 2017). We therefore consider existing POS taggers for tweets instead of developing our own. On the annotation scheme designed in §2.3, based on UD and adapted for Twitter, we compared several existing systems: the Stanford CoreNLP tagger, Owoputi et al.'s (2013) word cluster–enhanced tagger (both greedy and CRF variants), and Ma and Hovy's (2016) neural network tagger which achieves the state-of-the-art performance on PTB. Gui et al. (2017) presented a state-of-the-art neural tagger for Twitter, but their implementation works only with the PTB tagset, so we exclude it. All compared systems were re-trained on the combination of the UD_English-EWT and TWEEBANK V2 training sets. We use Twitter-specific GloVe embeddings released by Pennington et al. (2014) in all neural taggers and parsers.[13]

Experimental results. We tested the POS taggers on the TWEEBANK V2 test set. Results with gold-standard tokenization are shown in Table 4. Careful feature engineering and Brown et al. (1992) clusters help Owoputi et al.'s (2013) feature-based POS taggers to outperform Ma and Hovy's (2016) neural network model.

[12] We use the updated version of Twokenizer from Owoputi et al. (2013).

[13] http://nlp.stanford.edu/data/glove.twitter.27B.zip

System	Accuracy
Stanford CoreNLP	90.6
Owoputi et al., 2013 (greedy)	93.7
Owoputi et al., 2013 (CRF)	94.6
Ma and Hovy, 2016	92.5

Table 4: POS tagger comparison on gold-standard tokens in the TWEEBANK V2 test set.

Tokenization System	F_1
Stanford CoreNLP	92.3
our bi-LSTM tokenizer (§3.1)	93.3

Table 5: Owoputi et al. (2013) POS tagging performance with automatic tokenization on the TWEEBANK V2 test set.

Results of the Owoputi et al. (2013) tagger with non-greedy inference on automatically tokenized data are shown in Table 5. We see that errors in tokenization do propagate, but tagging performance is above 93% with our tokenizer.

3.3 Parser

Social media applications typically require processing large volumes of data, making speed an important consideration. We therefore begin with the neural greedy stack LSTM parser introduced by Ballesteros et al. (2015), which can parse a sentence in linear time and harnesses character representations to construct word vectors, which should help mitigate the challenge of spelling variation. We encourage the reader to refer their paper for more details about the model.

In our initial experiments, we train our parser on the combination of UD_English-EWT and TWEEBANK V2 training sets. Gold-standard tokenization and automatic POS tags are used. Automatic POS tags are assigned with 5-fold jackknifing. Hyperparameters are tuned on the TWEEBANK V2 development set. Unlabeled attachment score and labeled attachment score (including punctuation) are reported. All the experiments were run on a Xeon E5-2670 2.6 GHz machine.

Reimers and Gurevych (2017) and others have pointed out that neural network training is nondeterministic and depends on the seed for the random number generator. Our preliminary experiments confirm this finding, with a gap of 1.4 LAS on development data between the best (76.2) and worst (74.8) runs. To control for this effect, we report the average of five differently-seeded runs, for each of our models and the compared ones.

System	UAS	LAS	Kt/s
Kong et al. (2014)	81.4	76.9	0.3
Dozat et al. (2017)	81.8	77.7	1.7
Ballesteros et al. (2015)	80.2	75.7	2.3
Ensemble (20)	83.4	79.4	0.2
Distillation ($\alpha = 1.0$)	81.8	77.6	2.3
Distillation ($\alpha = 0.9$)	82.0	77.8	2.3
Distillation w/ exploration	82.1	77.9	2.3

Table 6: Dependency parser comparison on TWEEBANK V2 test set, with automatic POS tags. We use Ballesteros et al. (2015) as our baseline and build the ensemble and distilling model over it. The "Kt/s" column shows the parsing speed evaluated by thousands of tokens the model processed per second.

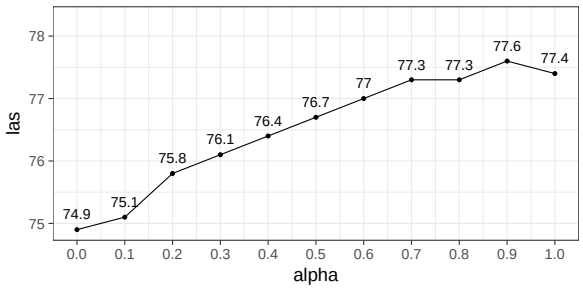

Figure 5: The effect of α on distillation.

Initial results. The first section of Table 6 compares the stack LSTM with TWEEBOPARSER (the system of Kong et al., 2014) and the state-of-the-art parser in the CoNLL 2017 evaluations, due to Dozat et al. (2017). Kong et al.'s (2014) parser is a graph-based parser with lexical features and word cluster and it uses dual decomposition for decoding. The parser in Dozat et al. (2017) is also a graph-based parser but includes character-based word representation and uses a biaffine classifier to predict if an attachment exists between two words. Both of the compared systems require superlinear runtime due to graph-based parsing. They are re-trained on the same data as our system. Our baseline lags behind by nearly two LAS points but runs faster than both of them.

Ensemble. Due to ambiguity in the training data—which most loss functions are not robust to (Frénay and Verleysen, 2014), including the log loss we use, following Ballesteros et al. (2015)—and due to the instability of neural network training, we follow Dietterich (2000) and consider an ensemble of twenty parsers trained using different random initialization. To parse at test time, the transition probabilities of the twenty members of the ensemble are averaged. The result achieves LAS of 79.4, outperforming all three systems above (Table 6).

Distillation. The shortcoming of the 20-parser ensemble is, of course, that it requires twenty times the runtime of a single greedy parser, making it the slowest system in our comparison. Kuncoro et al. (2016) proposed the distillation of 20 greedy transition-based parser into a single *graph-based* parser; they transformed the votes of the ensemble into a structured loss function. However, as Kuncoro et al. pointed out, it is not straightforward to use a structured loss in a *transition-based* parsing algorithm. Because fast runtime is so important for NLP on social media, we introduce a new way to distill our greedy ensemble into a single transition-based parser (the first such attempt, to our knowledge).

Our approach applies techniques from Hinton et al. (2015) and Kim and Rush (2016) to parsing. Note that training a transition-based parser typically involves the transformation of the training data into a sequence of "oracle" state-action pairs. Let $q(a \mid s)$ denote the distilled model's probability of an action a given parser state s; let $p(a \mid s)$ be the probability under the ensemble (i.e., the average of the 20 separately-trained ensemble members). To train the distilled model, we minimize the interpolation between their distillation loss and the conventional log loss:

$$\begin{aligned}\operatorname{argmin}_q \quad & \alpha \sum_i \underbrace{\sum_a -p(a \mid s_i) \cdot \log q(a \mid s_i)}_{\text{distillation loss}} \\ & + (1-\alpha) \sum_i \underbrace{-\log q(a_i \mid s_i)}_{\text{log loss}}\end{aligned} \tag{1}$$

Distilling from this parser leads to a single greedy transition-based parser with 77.8 LAS—better than past systems but worse than our more expensive ensemble. The effect of α is illustrated in Figure 5; generally paying closer attention to the ensemble, rather than the conventional log loss objective, leads to better performance.

Learning from exploration. When we set $\alpha = 1$, we eliminate the oracle from the estimation procedure (for the distilled model). This presents an opportunity to learn with *exploration*, by randomly sampling transitions from the ensemble, found useful in recent methods for training greedy

Pipeline stage	Score	Ours	SOTA
Tokenization	F_1	98.3	97.3
POS tagging	F_1	93.3	92.2
UD parsing	*LAS* F_1	74.0	71.4

Table 7: Evaluating our pipeline against a state-of-the-art pipeline.

models that use dynamic oracles (Goldberg and Nivre, 2012, 2013; Kiperwasser and Goldberg, 2016; Ballesteros et al., 2016). We find that this approach outperforms the conventional distillation model, coming in 1.5 points behind the ensemble (last line of Table 6).

Pipeline evaluation. Finally, we report our full pipeline's performance in Table 7. We also compare our model with a pipeline of the state-of-the-art systems (labeled "SOTA"): Stanford CoreNLP tokenizer,[14] Owoputi et al.'s (2013) tagger, and Dozat et al.'s (2017) parser. Our system differs from the SOTA pipeline in the tokenization and parser components. From Table 7, our pipeline outperforms the SOTA when evaluated in pipeline manner. The results also emphasize the importance of tokenization: without gold tokenization UD parsing performance drops by about four points.

4 Conclusion

We study the problem of parsing tweets into Universal Dependencies. We adapt the UD guidelines to cover special constructions in tweets and create the TWEEBANK V2, which has 55,607 tokens. We characterize the disagreements among our annotators and argue that inherent ambiguity in this genre makes consistent annotation a challenge. Using this new treebank, we build a pipeline system to parse tweets into UD. We also propose a new method to distill an ensemble of 20 greedy parsers into a single one to overcome annotation noise without sacrificing efficiency. Our parser achieves an improvement of 2.2 in LAS over a strong baseline and outperforms other state-of-the-art parsers in both accuracy and speed.

Acknowledgments

We thank Elizabeth Clark, Lucy Lin, Nelson Liu, Kelvin Luu, Phoebe Mulcaire, Hao Peng, Maarten Sap, Chenhao Tan, and Sam Thomson at the University of Washington, and Austin Blodgett, Lucia Donatelli, Joe Garman, Emma Manning, Angela Yang, and Yushi Zhang at Georgetown University for their annotation efforts in the first round. We are grateful for the support from Lingpeng Kong at the initial stage of this project. We also thank the anonymous reviewers for their helpful comments and suggestions. This work was supported by the National Key Basic Research Program of China via grant 2014CB340503 and the National Natural Science Foundation of China (NSFC) via grant 61632011.

[14] We choose the Stanford CoreNLP tokenizer in the spirit of comparing rule-based and statistical methods.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2016. Many languages, one parser. *TACL* 4.

Miguel Ballesteros, Chris Dyer, and Noah A. Smith. 2015. Improved transition-based parsing by modeling characters instead of words with LSTMs. In *Proc. of EMNLP*.

Miguel Ballesteros, Yoav Goldberg, Chris Dyer, and Noah A. Smith. 2016. Training with exploration improves a greedy stack LSTM parser. In *Proc. of EMNLP*.

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).

Marie-Catherine de Marneffe, Timothy Dozat, Natalia Silveira, Katri Haverinen, Filip Ginter, Joakim Nivre, and Christopher D. Manning. 2014. Universal Stanford Dependencies: A cross-linguistic typology. In *LREC*.

Leon Derczynski, Alan Ritter, Sam Clark, and Kalina Bontcheva. 2013. Twitter part-of-speech tagging for all: Overcoming sparse and noisy data. In *Proc. of RANLP 2013*.

Thomas G. Dietterich. 2000. An experimental comparison of three methods for constructing ensembles of decision trees: Bagging, boosting, and randomization. *Machine Learning* 40(2):139–157. https://doi.org/10.1023/A:1007607513941.

Timothy Dozat, Peng Qi, and Christopher D. Manning. 2017. Stanford's graph-based neural dependency parser at the CoNLL 2017 shared task. In *Proc. of CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

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

Jennifer Foster, Özlem Çetinoğlu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef Van Genabith. 2011. #hardtoparse: POS tagging and parsing the Twitterverse. In *Proc. of the 5th AAAI Conference on Analyzing Microtext*.

Benoît Frénay and Michel Verleysen. 2014. Classification in the presence of label noise: A survey. *IEEE Transactions on Neural Networks and Learning Systems* 25:845–869.

Kim Gerdes. 2013. Collaborative dependency annotation. In *Proc. of DepLing*.

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*.

Yoav Goldberg and Joakim Nivre. 2012. A dynamic oracle for arc-eager dependency parsing. In *Proc. of COLING*.

Yoav Goldberg and Joakim Nivre. 2013. Training deterministic parsers with non-deterministic oracles. *TACL* 1.

Tao Gui, Qi Zhang, Haoran Huang, Minlong Peng, and Xuanjing Huang. 2017. Part-of-speech tagging for Twitter with adversarial neural networks. In *Proc. of EMNLP*.

Jiang Guo, Wanxiang Che, David Yarowsky, Haifeng Wang, and Ting Liu. 2015. Cross-lingual dependency parsing based on distributed representations. In *Proc. of ACL*.

Geoffrey E. Hinton, Oriol Vinyals, and Jeffrey Dean. 2015. Distilling the knowledge in a neural network. *CoRR* abs/1503.02531.

Zhiheng Huang, Wei Xu, and Kai Yu. 2015. Bidirectional LSTM-CRF models for sequence tagging. *CoRR* abs/1508.01991.

Yoon Kim and Alexander M. Rush. 2016. Sequence-level knowledge distillation. In *Proc. of EMNLP*.

Eliyahu Kiperwasser and Yoav Goldberg. 2016. Simple and accurate dependency parsing using bidirectional LSTM feature representations. *TACL* 4.

Lingpeng Kong, Nathan Schneider, Swabha Swayamdipta, Archna Bhatia, Chris Dyer, and Noah A. Smith. 2014. A dependency parser for tweets. In *Proc. of EMNLP*.

Adhiguna Kuncoro, Miguel Ballesteros, Lingpeng Kong, Chris Dyer, and Noah A. Smith. 2016. Distilling an ensemble of greedy dependency parsers into one MST parser. In *Proc. of EMNLP*.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional LSTM-CNNs-CRF. In *Proc. of ACL*.

Christopher D. Manning, Mihai Surdeanu, John Bauer, Jenny Finkel, Steven J. Bethard, and David McClosky. 2014. The Stanford CoreNLP natural language processing toolkit. In *ACL System Demonstrations*.

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

Joakim Nivre, Marie-Catherine de Marneffe, Filip Ginter, Yoav Goldberg, Jan Hajic, Christopher D. Manning, Ryan McDonald, Slav Petrov, Sampo Pyysalo, Natalia Silveira, Reut Tsarfaty, and Daniel Zeman. 2016. Universal Dependencies v1: A multilingual treebank collection. In *Proc. of LREC*.

Brendan O'Connor, Michel Krieger, and David Ahn. 2010. TweetMotif: Exploratory search and topic summarization for Twitter.

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*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proc. of EMNLP*.

Slav Petrov, Dipanjan Das, and Ryan McDonald. 2012. A universal part-of-speech tagset. In *Proc. of LREC*.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of LSTM-networks for sequence tagging. 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*.

Manuela Sanguinetti, Cristina Bosco, Alessandro Mazzei, Alberto Lavelli, and Fabio Tamburini. 2017. Annotating Italian social media texts in Universal Dependencies. In *Proc. of Depling*. Pisa, Italy.

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*.

Milan Straka and Jana Straková. 2017. Tokenizing, POS tagging, lemmatizing and parsing UD 2.0 with UDPipe. In *Proc. of the CoNLL 2017 Shared Task: Multilingual Parsing from Raw Text to Universal Dependencies*.

Hongmin Wang, Yue Zhang, GuangYong Leonard Chan, Jie Yang, and Hai Leong Chieu. 2017. Universal Dependencies parsing for colloquial Singaporean English. In *Proc. of ACL*.

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