Parsing Tweets into Universal Dependencies

Anonymous ACL submission

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. Using the extended guidelines [including tokenization, POS tagging and parsing? (need to mention the annotation process/pipeline system not only consists parsing in abstract) —YI], we create a new tweet treebank (TWEEBANK V2), which has 3,550 tweets[add tweet # —YI][YL didn't see necessity of mentioning # of tweets] and 55,607 tokens with labeled attachments. It is more than four times larger than the (unlabeled) TWEEBANK V1 introduced by Kong et al. (2014). We characterize the disagreement among our annotations and show that it is challenging to deliver consistent annotation due to ambiguity in understanding and explaining tweets. Using the new treebank, we build a pipeline system to parse raw tweets into UD. To overcome the 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 2.6 in LAS over [which comparison is this? —NAS][YL my bad, it was fixed.] the 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, both because domain adaptation is required, but 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 (Marcus et al., 1993, PTB). Stanford dependencies were converted afterwards in their work [by who? —NAS][YL by themselves]. 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 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. Our annotation includes tokenization, part-of-speech (POS) tags, and (labeled) universal dependencies. We characterize the disagreement among our annotations [annotation(s) and disagreement(s). check abstract, same sentence, different singularity/plurality. I think should be 'disagreement' and 'annotations' —YI] and find that consistent annotation [annotations? —YI] is still challenging to deliver even with the extended guidelines.

Based on these annotations, we 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" 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 2.6 points [again, not sure what this is quoting —NAS][$^{Y}_{L}$ fixed] in LAS over the baseline parser can be achieved with our distillation method and it outperforms other state-of-the-art parsers in both accuracy and running speed.

The contributions of this paper include:

- We study the challenges of annotating tweets in UD (§2) and created a new tweet treebank (TWEEBANK V2), which includes tokenization, part-of-speech tagging, and labeled universal dependencies annotation. We also characterize the difficulties of creating such annotation.
- We introduce and evaluate a [pipeline —YI] system to parse the raw tweet text into universal dependencies (§3). Experimental results show it performs better than a stack of the state-of-the-art alternatives. [say it performs better than existing systems? —NAS][$^{Y}_{L}$ added.]
- We propose a new distillation method for training a greedy parser, leading to better performance than existing methods and without efficiency sacrifices.

We release our dataset and our system as open-source software at `http://anonymized`.

2 Annotation

We first review TWEEBANK V1 of Kong et al. (2014), which is [was? —YI] the largest [unlabelled? —YI] Twitter dependencies annotation (§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 (§2.5).

2.1 Background: TWEEBANK

The annotation effort we describe stands in contrast to [the? —YI] 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 [unlabelled? —YI] 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 mostly by nonnative speakers [should not emphasize non-native speakers (v2 also built by non-native speakers), but emphasize annotators with only cursory training, no clear annotation guidelines, no consensus/agreement on controversial cases, allow annotators to underspecify part of the tree and allow multiple different trees for the same tweets —YI], 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 significant [significant? —YI] 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,[1] 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)[are words like gonna called contractions, or do they have another term? —YI]. But acronyms are not touched in the UD tokenization guidelines. Thus, we follow the UD tokenization for contractions and left acronyms like *idc* as a single token. [unless we make a change, I think we need to say here that acronyms like *idc* are left as a single token —NAS] [$^{Y}_{L}$mentioned acronymns, we don't have the problem of normalization, cause normalization was just annotated for data study.]

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 simplification. Such tokens only account for 0.067% 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? —YI], for consistency with other UD efforts, which adopt the universal POS tagset of Petrov et al. (2012). In some cases, conflicts arose between the UD English treebank conventions (de Marneffe et al., 2014, UD_English)[2] and the conventions of Gimpel et al. (2011) and Owoputi et al. (2013). [The conflicts should refer to a **noncorresponding tag** from that of gimpel to UDEnglish, cuz they dont have the same tagset. —YI] In these cases, we always conformed to UD, enabling consistency (e.g., when we exploit the existing UD_English 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.

Tokens that abbreviate multiple words, such as *idc* ("I don't care") 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. (2010)'s tokenization, Gimpel et al. introduced new parts of speech for these cases instead.[3] 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

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

[2] `https://github.com/UniversalDependencies/UD_English`

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

	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.

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. How should we syntactically analyze acronym tokens like *idc* (abbreviating "I don't care") and *rn* ("right now")? Should they be decomposed into their component words, and if so should those words be "normalized" into explicitly spelled out intermediate forms? [this sentence should be deleted or moved to tokenization? —YI] [As we already know from previous sections, *idc* is not tokenized and tagged as verb, we follow ... —YI] [Normalization is discussed in the 2.5.2 Annotation Process. Actually we did do normalization, but just as auxiliary info, didn't use it for any model. —YI][$^{\text{Y}}_{\text{L}}$i think acronym is a different story to tokenization.] We follow Kong et al. (2014) and annotate their syntax 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 is an unnecessarily complicated step.[Again we did normalization! —YI]

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: sentiment emoticons, retweet markers and their following at-mentions, topical hashtags, referential URLs, and truncated words.[4] 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 non-syntactic; detailed statistics can be found in Table 1. [$^{\text{Y}}_{\text{L}}$I've merged retweet marker and retweet at-mention into one.]

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`, and emoticons and hashtags may be used as content words within a sentence[at-mentions can be a normal vocative proper noun —YI]; 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:[shall we mention how we segment sentences? Seems relevant here —YI][$^{\text{Y}}_{\text{L}}$i don't want to touch sentence segmentation, it will makes the paper more messy.]

- If one syntactic token is within a sentence that has a clear predicate, it will be attached to this predicate. Retweet construction is a special case and we will discuss our choice in the following paragraph;
- If the whole sentence is made of 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 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 dependencies annotation. In order to keep our annotation conventions in line with UD norms [and preserve the original tweets as much as possible —YI], 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.

[4]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.

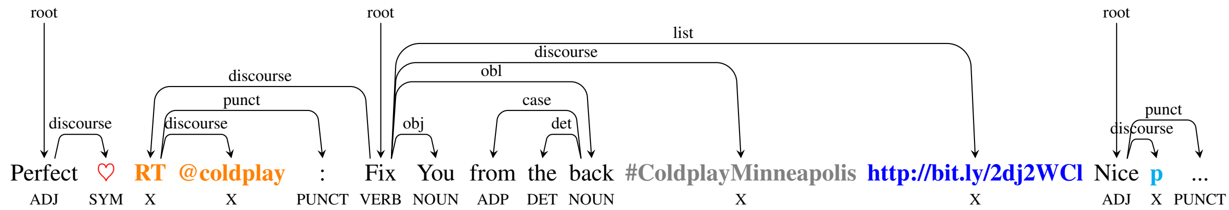

Figure 1: An example tweet that contains non-syntactic tokens: sentiment emoticon, retweet marker and its following at-mention, topical hashtag, referential URL, and truncated word. This example tweet is a concatenation of three real tweets.

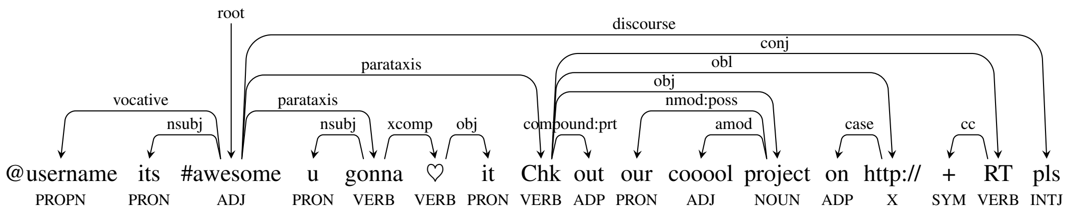

Figure 2: An example tweet with informal but syntactic tokens. This is a contrived example inspired by several tweets.

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 its object[or might be treated as 2 non-syntactic tokens and a punctuation, which should all attach to the predicate of the following retweeted tweet. —YI][$^{Y}_{L}$i wouldn't like the readers to think in this direction.]. This solution would lead to an uninformative root word 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 well handled by UD conventions: ellipsis, irregular word orders, and paratactic sentences [or phrases/components —YI] not explicitly delineated by punctuation.

Vocative at-mentions. Another idiomatic construction on Twitter is an at-mention as a sign that a tweet is a reply to a tweet by the mentioned user[not necessary, can be just mentioning somebody. should be careful to say it. We can say this construction include 1. reply; 2. mention sb. Both cases are similar to general vocative proper noun case in standard texts. And we need to exclude at-mentions in Retweet Construction. —YI][$^{Y}_{L}$i didn't see any difference between reply and mention, they all starts with a @username.]. We treat these at-mentions as vocative expressions, labeling them as *proper noun* (PROPN)[here is confusing that PROPN could be thought as a label in dependencies. —YI] and attaching them to the following predicate with the label *vocative* (see Figure 2 for an example) [just as UD guidelines —YI].

2.5 TWEEBANK V2

Following the guidelines presented above, we create a new annotated[del? —YI] Twitter [Twitter dependency? —YI] treebank, which we call TWEEBANK V2.

2.5.1 Data Collection

TWEEBANK V2 is built on the original data of TWEEBANK V1 (840 unique tweets, 639/201 for training/test set), 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.[5] The latter data source consists of 147.4M English tweets after being filtered by the

[5]Data downloaded from `https://archive.org/`.

	train	dev	test
tweets	1,639	709	1,201
	+1,000	+709	+1,000
tokens	24,753	11,742	19,112

Table 2: Statistics of TWEEBANK V2. The second column shows the number of new tweets compared to TWEEBANK V1.

lang attribute in the tweet JSON and *langid.py* toolkit.[6]

2.5.2 Annotation Process

Our annotation process was conducted in three stages. In the first stage, 18 researchers worked on the TWEEBANK V1 proportion and created the initial annotations in one day. Before annotating, they were given a tutorial overview of the [general —YI] UD annotation conventions and our guidelines [specifically for annotating tweets —YI]. 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 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. Finally, an extra layer of word-level normalization[see 10 th col of conllu data, also mention we could recover the original tweets also from this col (tokenization) —YI][$^{Y}_{L}$didn't get your point. if mentioning the normalization here introduce too much confusions, we would better remove these.] was manually annotated for data analysis purposes. For example, *idc* is annotated as 'I_don't_care' and *4 my FAM* is as 'for my familiy'. Such normalization doesn't change our tokenization annotation. [should mention tokenization process, say we tokenize V1 and train tokenizer for V2 data, then manually correct them. —YI][$^{Y}_{L}$add "a tokenizer"] [add final data split —YI][$^{Y}_{L}$it's better to use table 2. what's more, i don't think we benefit from re-stating our data source.] We obtained 3,550 labelled annotated tweets, split into train/dev/test set with /(639 + 1000)/(210 + 500)/(201 + 1000) tweets, where each set consists of both old tweets from Gimpel et al. (2011) and new tweets from the Twitter stream. [should also mention main annotators (authors) are non-native English speakers? —YI]

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

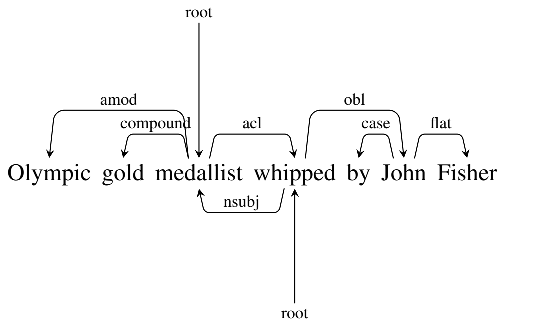

Figure 3: An example of disagreement; one annotator's parse is shown above, disagreeing arcs from the other annotator are shown below. This is a real example in our annotation.

We report the inter-annotator agreement between the annotators in the second stage. There is few disagreement on the tokenization annotation. The agreement on POS is 96.6%, the unlabeled dependency agreement is 88.8% and the labeled dependency agreement is 84.3%. Further analysis shows the major disagreements on POS involve entity names (30.6%) [noun phrases or proper noun phrases, do we need example here? —YI][$^{Y}_{L}$i wasn't counting PROPN and NOUN confusion, but any type of confusion related with PROPN] and topical hashtags (18.1%). Taking the example in Figure 1, "Fix you" [need to relate to previous error analysis, entity names maybe? —YI]can be understood as a verbal phrase but also as the name of the Coldplay's single and tagged as proper noun. [need to say the reason why dependency inter-annotator agreement is low? 1. POS disagreement lead to big difference in understanding the semantics. 2. Some special constructions like ellipsis lead to different view of tweets. Fig3 is an example of 2. —YI]An example of a disagreement on dependencies is shown in Figure 3. Depending on whether this is an example of an empty auxiliary verb, or a clause-modified noun, either annotation is plausible.

3 Pipeline

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

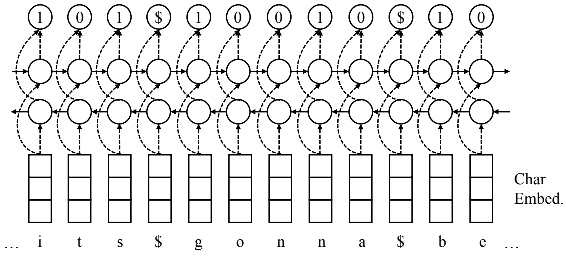

Figure 4: The bi-LSTM tokenizer model that segments 'its gonna be' into 'it s gon na be'.

System	*F1*
Stanford CoreNLP	96.6
Twokenizer	94.4
UD pipe v1.2	97.3
our bi-LSTM tokenizer	**98.3**

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

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 for 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 our tokenization model. Space is treated as an input but consistently assigned a special tag $.

Experimental results. We trained our tokenizer on the training portion[set? —YI] of TWEEBANK V2 combined with the UD_English training dataset 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) systems are rule-based systems and were not adapted to the UD tokenization scheme. The UD pipe v1.2 (Straka and Straková, 2017) model was re-trained on the same data as our system. Compared with the UD pipe, we use LSTM instead of GRU in our model and we also use a larger size of hidden units (64 against 20), which has stronger representation power. [we need to say what their model was; why does ours work better? from their paper, it seems that they used a GRU instead of an LSTM ... is that all? do we win because of better tuning, or what? —NAS][$^{Y}_{L}$done.] Our bi-LSTM tokenizer achieves the best accuracy among all these tokenizers. These results indicate the value of statistical modeling in tokenization for informal texts.

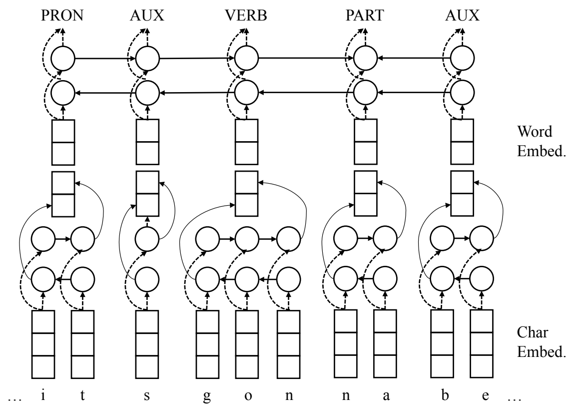

Figure 5: The bi-LSTM POS tagger model that tags 'it s gon na be' as `PRON AUX VERB PART AUX`.

3.2 Part-of-Speech Tagger

Part-of-speech tagging for tweets has been extensively studied (Ritter et al., 2011; Gimpel et al., 2011; Owoputi et al., 2013; Gui et al., 2017). On the annotation scheme designed in §2.3, based on UD and adapted for Twitter, we compared several existing systems and also two bi-LSTM systems similar to Huang et al. (2015) and Lample et al. (2016), one using word vectors and the other using word and character vectors (see Figure 5). All systems were re-trained on the combination of the UD_English and TWEEBANK V2 training sets. We use Twitter-specific glove embeddings released by Pennington et al. (2014) in all neural taggers and parsers.[7]

Experimental results. We tested the POS taggers on the TWEEBANK V2 test set. Results with gold-standard tokenization are shown in Table 4. The careful feature engineering in the POS tagger of Owoputi et al. (2013) outperforms our neural

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

System	*Precision*
Stanford CoreNLP	90.4
Owoputi et al., 2013 (greedy)	94.2
Owoputi et al., 2013 (CRF)	**95.1**
Ma and Hovy, 2016	91.8
our word bi-LSTM	89.2
our character + word bi-LSTMs	91.5

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

System	*F1*
Stanford CoreNLP	92.1
our bi-LSTM tokenizer (§3.1)	93.6

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

network models. (That tagger also makes use of Brown et al. (1992) clusters derived from a large collection of unannotated tweets. Those clusters did not improve the performance of our bi-LSTM models.)

Results of the Owoputi et al. (2013) 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 desideratum. We therefore begin with the neural greedy stack LSTM parser introduced by Ballesteros et al. (2016), which can parse a sentence in linear time and harnesses character representations, 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 preliminary experiments, we train our parser on the combination of UD_English 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 ran on a Xeon E5-2670 2.6 GHz machine.

System	UAS	LAS	SP
Kong et al. (2014)	81.6	77.2	0.3
Dozat and Manning (2016)	81.9	77.7	1.7
Ballesteros et al. (2016)	80.4	75.8	2.3
Ensemble (20)	83.5	79.4	0.2
Distillation ($\alpha = 1.0$)	82.2	78.0	2.3
Distillation ($\alpha = 0.9$)	82.4	78.1	2.3
Distillation w/ exploration	82.5	78.4	2.3

Table 6: Dependency parser comparison on TWEEBANK V2, with automatic POS tags. The *SP* column shows the parsing speed evaluated by the number of thousand tokens the model processed per second. For fair comparison, we limit the number of CPU used in Dozat and Manning (2016) experiments to 1.

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 LAS on development data between the best (75.8) 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. [please check! we did this for all systems, not just ours, right? —NAS][$^{Y}_{L}$yes. added]

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 and Manning (2016), both of which are graph-based parsers requiring superlinear runtime. [can we quantify how much time each one takes, either in tweets parsed per second or seconds per tweet? —NAS][$^{Y}_{L}$done.] Both of the comparison systems are re-trained on the same data as our system. Our system lags behind by nearly two LAS points but runs faster than both of them.

Ensemble. Due both[? —YI] 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 re-

Pipeline	Metrics	ours	SOTA stack
Tokenization	*F1*	98.3	96.6
POS tagging	*Precision F1*	93.7	92.1
Universal dependencies	*LAS F1*	74.1	70.3

Table 7: Evaluating our model in pipeline manner. [should this be "F1"? shouldn't the last line be LAS? —NAS][YL added metrics column]

sult achieves LAS of 79.4, outperforming all three systems above (Table 6). However, ensembling 20 parsers also significantly slows down the parsing speed and leads to the slowest system in our comparison.

Distillation. The shortcoming of the 20-parser ensemble is, of course, that it requires twenty times the runtime of a single greedy parser. 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 follows Hinton et al. (2015) and Kim and Rush (2016). 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) [check —NAS] [YL checked]. To train the distilled model, we minimize the interpolation between their distillation loss and the conventional log loss:

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

Distilling from this parser leads to a single greedy transition-based parser with 78.1 LAS—better than past systems but worse than our more expensive ensemble. The effect of α is illustrated in Figure 6; generally paying closer attention to the ensemble, rather than the conventional log loss objective, leads to better performance.

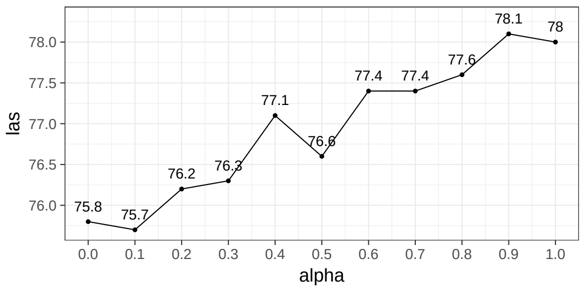

Figure 6: The effect of α on distillation.

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, [we? —YI] found useful in recent methods for training greedy models that use dynamic oracles (Ballesteros et al., 2016). We find that this approach outperforms the conventional distillation model, coming in only one point 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 stack of the state-of-the-art systems (namely, the *SOTA stack*): Stanford CoreNLP tokenizer[why not compare with UD pipe? —YI][YL i was thinking stanford tokenizer is more suitable, but i will get the udpipe result], Owoputi et al. (2013)'s tagger, and Dozat and Manning (2016)'s parser. Our system differs with the SOTA stack in the tokenization and parser components. From Table 7, our system outperforms the SOTA stack when evaluated in pipeline manner. The results also emphasize the importance of segmentation[you mean word tokenization or sentence segmentation? —YI][YL we didn't touch sentence seg.]: without gold segmentation, UD parsing performance drops by more than 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 3,550 tweets[add tweet # —YI] and 55,607 tokens. We characterize the disagreements among our annotations and argue that inherent ambiguity in this genre makes consistent annotaiton 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 2.6 [which comparison are you quoting here? —NAS][$^{Y}_{L}$ fixed] in LAS over a strong baseline and outperforms other state-of-the-art parsers in both accuracy and speed.

References

Waleed Ammar, George Mulcaire, Miguel Ballesteros, Chris Dyer, and Noah 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. *Comput. Linguist.* 18(4). http://dl.acm.org/citation.cfm?id=176313.176316.

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*.

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 and Christopher D. Manning. 2016. Deep biaffine attention for neural dependency parsing. *CoRR* abs/1611.01734. http://arxiv.org/abs/1611.01734.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proc. of NAACL*. http://www.aclweb.org/anthology/N13-1037.

Jennifer Foster, Ozlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Stephen Hogan, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. #hardtoparse: Pos tagging and parsing the twitterverse.

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.

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*. ACL.

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-2017*. ACL. https://www.aclweb.org/anthology/D17-1255.

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. http://arxiv.org/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-2016*. ACL. https://aclweb.org/anthology/D16-1139.

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

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*.

Guillaume Lample, Miguel Ballesteros, Sandeep Subramanian, Kazuya Kawakami, and Chris Dyer. 2016. Neural architectures for named entity recognition. In *Proc. of NAACL*. ACL. http://www.aclweb.org/anthology/N16-1030.

Xuezhe Ma and Eduard Hovy. 2016. End-to-end sequence labeling via bi-directional lstm-cnns-crf. In *Proc. of ACL*. ACL. http://www.aclweb.org/anthology/P16-1101.

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*. http://www.aclweb.org/anthology/P/P14/P14-5010.

Mitchell P. Marcus, Beatrice Santorini, and Mary Ann Marcinkiewicz. 1993. Building a large annotated corpus of english: The penn treebank. *Computational Linguistic* 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 2016*. ELRA.

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*. ACL.

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*. ELRA.

Nils Reimers and Iryna Gurevych. 2017. Reporting score distributions makes a difference: Performance study of lstm-networks for sequence tagging. In *Proc. of EMNLP-2017*. https://www.aclweb.org/anthology/D17-1035.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proc. of EMNLP-2011*. ACL. http://www.aclweb.org/anthology/D11-1141.

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*. Association for Computational Linguistics, Sofia, Bulgaria, pages 51–60. http://www.aclweb.org/anthology/W13-2307.

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*. ACL.

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*. ACL, pages 1732–1744. http://aclweb.org/anthology/P17-1159.

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