Parsing Tweets into Universal Dependencies

Anonymous ACL submission

Abstract

In this paper, we study the problem of analyzing tweets with universal dependencies (Nivre et al., 2016, UD). We extend the UD guidelines to cover special constructions in tweets. Under such guidelines, we create a new tweet treebank (TWEEBANK V2), which has 55,607 tokens with labelled attachments.[labelled attachments —YI] It is more than 4 times larger than the unlabelled TWEEBANK V1 (Kong et al., 2014). [unlabelled attachments for the original Tweebank —YI] We characterize the disagreements among our annotation and show that it is challenging to deliver consistent annotation due to the ambiguities in ways of understanding and explaining the tweets. Over the new treebank, we build a pipeline system to parse raw tweets into UD. To conquer the annotation noise without sacrificing efficiency, we propose a new method to distill an ensemble of 20 parsers into a single one. Our parser achieves an improvement of 2.3 in LAS over the baseline and outperforms other state-of-the-art parsers.

1 Introduction

Analyzing the syntax of tweets is challenging. The challenges not only come from the difficulty of adapting the parser trained on the standard text to the Twitter domain, but also comes from creating reasonable dataset for training and evaluating tweet parsers. Foster et al. (2011) pioneered in this research direction by annotating Penn Treebank (Marcus et al., 1993, PTB) constituencies to a set of tweets which contains 7,630 tokens. Stanford dependencies were converted afterwards. Kong et al. (2014) further studied the challenges in annotating tweets and presented a tweet treebank (TWEEBANK) which has 12,149 tokens and largely followed the conventions suggested by Schneider et al. (2013) that are close to Yamada and Matsumoto (2003) dependencies. Both annotation efforts were highly influenced by the PTB, whose guideline has a good grammatical coverage on newswire. However, when it comes to the noisy and informal user generated text, it's questionable whether such good coverage holds.

Universal dependencies (Nivre et al., 2016, UD) was created to deliver consistent annotation across different languages. To allow such consistency, UD was designed to be adaptive to different genres and languages (Guo et al., 2015; Ammar et al., 2016). It's promising that analyzing the syntax of tweets can benefit from such adaptability. In this paper, we create a new tweet treebank of 55,607 tokens by following the UD guidelines but also contending the domain-specific challenges which were not covered by UD guidelines. Our annotation includes a whole pipeline of tokenization, part-of-speech (POS) tags and universal dependencies. We characterize the disagreement among our annotations and show that consistent annotation is still challenging to deliver even with the extended guidelines.

Based on these annotations, we built up a whole pipeline to parse the raw tweet text into universal dependencies. Our pipeline includes: a bidirectional LSTM (bi-LSTM) tokenizer, a word cluster enhanced POS tagger (Owoputi et al., 2013), and stack-lstm parser with character representation (Ballesteros et al., 2015).

To conquer the noise in our annotation and achieve better performance without sacrificing efficiency, we propose a new method to train the parser with the distillation (Hinton et al., 2015) of 20 parsers ensemble. We show that learning from the exploration of the ensemble parser can be more beneficial than just learning from the gold standard transition sequence in the sense of training a transition-based distilling parser. Experimental results show that an improvement of more than 2.3 points in LAS over baseline parser can be achieved with our distillation method and it outperforms other state-of-the-art parsers.

Contribution of this paper includes:

- We studied the challenges of annotating tweets in UD (§2) and created a new tweet treebank (TWEEBANK V2), which includes tokenization, part-of-speech tagging, and dependencies annotation. We also characterized the difficulties of creating such annotation.

- We built up a whole pipeline to parse the raw tweet text into universal dependencies (§3). We proposed a new distillation method in training the parser in our pipeline which achieve 2.3 points improvement in LAS without sacrificing efficiency and outperformed other state-of-the-art parsers in our comparison. [change verbs to past tense —YI]

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

[I didn't do anything significant to the intro. for every example, make clear whether it's "real" or contrived. —NAS] [$^{Y}_{L}$I've add the source of our example in the figure caption.]

2 Annotation

[Maybe we should contrast UD and kong's annotation strategy during each stage, because UD has also annotation guideline for every stage. —YI] In this section, we first review TWEEBANK V1 of Kong et al. (2014), which is the largest twitter dependencies annotation in previous literals (§2.1). Then we introduce the difference in our tokenization and part-of-speech (re)annotation with O'Connor et al. (2010) and Gimpel et al. (2011) on which TWEEBANK V1 was built (§2.3). [add tok —YI] Then, we move to our effort of adopting the UD conventions to cover tweet specific constructions. At last, we describe our process of creating new tweet treebank and characterize the difficulties in reaching consistent annotations. [need to introduce this section and explain the structure. —NAS] [$^{Y}_{L}$done.]

2.1 Background: TWEEBANK

The annotation effort we describe stands in contrast to 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 929 tweets. Their annotations added dependencies to a portion of the data annotated with POS tags by Gimpel et al. (2011) and Owoputi et al. (2013) after tokenization (O'Connor et al., 2010) [add tok —YI]. They also contributed a system for parsing; we defer the discussion of their parser to §3.

Kong et al. (2014)'s rapid, small-scale annotation effort was heavily constrained. It was carried out mostly by non-native speakers, 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 is in the middle of the strategy of O'Connor et al. (2010) and that of UD. We want to fit Twitter texts better in UD dependencies while preserving the original tweets as much as possible.

O'Connor et al. (2010) preserve most of non-informal tokens including the hashtags, at-mentions, emoticons & emojis and unicode glyphs. They also treat contractions and acronyms as whole tokens and do not split them. UD tokenization[1] , in order to serve better dependency an-

[1] http://universaldependencies.org/u/overview/tokenization.html[is it ok? no paper reference but webpage reference —YI]

notation, treat each syntactic word as token. They take more aggressive strategy, including splitting off clitics from contractions or multi-word expressions (*gonna*) and combine several orthographic tokens into a single syntactic word. [does UD support normalization for acronyms? Did not see it from website, need to check. related to "idc" —YI]

We do not touch any non-informal token in tweets and leave them as in the original tweets. Then we mainly follow UD tokenization. We tokenize contractions and multi-word expressions into their syntactic components, but do not combine tokens that have been already split in the original tweets back anymore. For example, words like *gonna* or *can't* will be tokenized to *gon na* and *ca n't*, and parts like *Y O* and *R E T W E E T* will be not connected back to single words[2].

2.3 Part-of-speech Annotation

Before turning to UD annotations, we (re)annotated the data with POS, 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) [3] and the conventions of Gimpel et al. (2011) and Owoputi et al. (2013). 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 (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 coarse-grained POS tagging 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 is in tokenization. UD calls for more aggressive tokenization than that used by Gimpel et al. (2011), which followed the rule-based system introduced by O'Connor et al. (2010). In particular, they opted out of tokenizing contractions and possessives, introducing new parts of speech instead.[4] 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.

[for each thing we talk about here, might be nice to quantify it in our annotated corpus —NAS]

Acronym abbreviations. How should we syntactically analyze acronym tokens like *idc* (abbreviating "I don't care") ['idc' should be discussed in tokenization section, and actually it is already implied from POS section that it should be treated as a single word. —YI] 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? 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.[again we should also mention UD tokenization, so might be worth adding a tokenization section —YI]

Eisenstein (2013) studied the necessity of normalization in social media text and pointed that such normalization is problematic. Our solution to the syntax of abbreviations follows the spirit of his argument. Because abbreviations which clearly

[2] Such tokens only accounts for 0.067%, and we use *goeswith* relation to resolve it.

[3] https://github.com/UniversalDependencies/UD_English

[4] 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: Proportion of non-syntactic tokens in the annotation.

carry syntactic functions only constitute 0.06% of the tokens in our dataset, we believe that normalization is an unnecessarily complicated step. [maybe we owe citations to (Finin et al., 2010; Eisenstein, 2013) here? maybe better papers to cite —NAS]

Non-syntactic tokens. The major characteristic that distinguishes tweets from standard texts is that there are large proportion of tokens that don't carry any syntactic function. In our annotation, there are five types of non-syntactic tokens: sentiment emoticons, retweet markers, topical hashtags, referential URLs, and truncated words [5]. 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, 7.5% of all tokens are non-syntactic; detailed statistics can be found in Table 1[6].

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; 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;
- 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, we include non-syntactic tokens in our annotation following the convention 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 its object. 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 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. We treat these at-mentions as vocative expressions, labeling them as *proper noun* (PROPN) and attaching them to the following predicate with the label *vocative* (see Figure 2 for an example).

2.5 TWEEBANK V2

[I think it should move to another section, just as in Kong's paper. —YI] Following the guidelines mentioned above, we create a new annotated Twitter treebank, which we call TWEEBANK V2.

[5] Tweets we analyze have 140 character limits. Although Twitter has now doubled the tweet limit to 280 characters, we believe this type of tokens will still remain.

[6] The statistics are obtained on 140 character limit tweets.

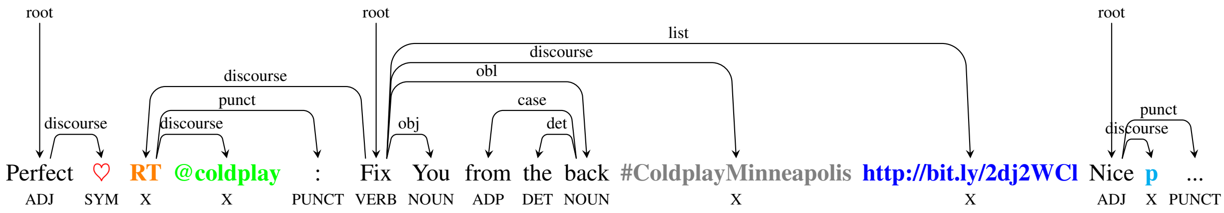

Figure 1: An example tweet that contains non-syntactic tokens: sentiment emoticon, retweet marker, retweet at-mention, topical hashtag, referential URL, and truncated word. This example tweet is a concatenation of three real tweets.

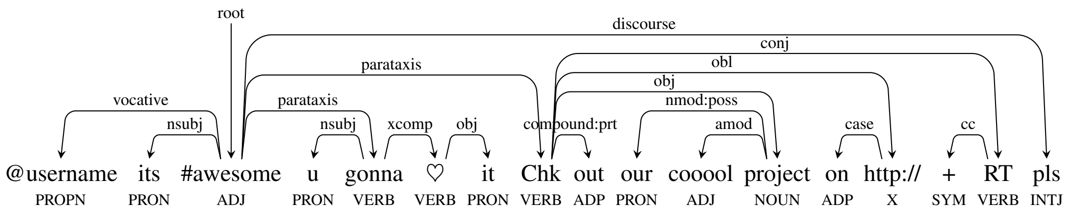

Figure 2: An example tweet with informal but syntactic tokens. This is an artificial tweets inspired by several tweets.

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 additional 210[changed to 210 —YI] tweets sampled from Gimpel et al. (2011) and 2,500 tweets sampled from the Twitter stream from February 2016 to July 2016[7]. The latter data source consists of 147.4M English tweets after being filtered by *lang* attribute in the tweet JSON and *langid.py*[8] toolkit.

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 UD annotation conventions and our guidelines. Both the guidelines and annotation were further refined by the authors of this paper to increase the coverage of our guideline and solve inconsistency between different annotators during this exercise. In the second stage, a POS tagger and parser were trained on the annotated data from the first stage (1,041 tweets in total), and used to automatically analyze the sampled 2,500 tweets. Authors of this paper manually corrected the parsed data. [this is a little weird: don't you have to do this before you parse? —NAS][$^{Y}_{L}$we were actually correcting the auto-analyzed 2,500 tweets.] Finally, an extra layer of word-level normalization was manually annotated for data analysis purposes.

[7]Data downloaded from https://archive.org/
[8]https://github.com/saffsd/langid.py

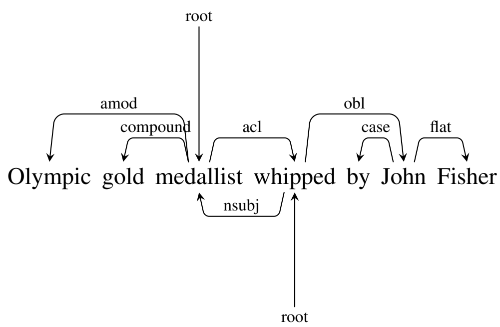

Figure 3: An example of disagreement; one annotator's parse is shown above, disagreeing arcs from the other annotator are shown below. This is a real example in our annotation.

We report the inter-annotator agreement between the annotators in the second stage. The agreement on POS is 96.61%, the unlabeled dependency agreement is 88.75% and the labeled dependency agreement is 84.31%. Further analysis shows the major disagreements on POS involve entity names (30.57%) and topical hashtags (18.11%). 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

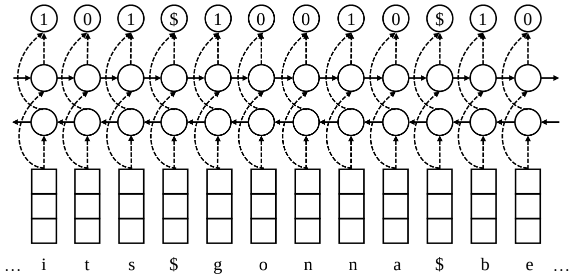

Figure 4: The bi-LSTM tokenizer model that segments 'its gonna be' into 'it s gon na be'.

System	F_1
Stanford CoreNLP	96.6
Twokenizer	94.4
UD pipe v1.2 (same training data)	97.3
Our bi-LSTM tokenizer	**98.3**

Table 2: Tokenizer comparison on the TWEEBANK V2 test set.

dependencies is shown in Figure 3. Depending on whether this is an example of an empty auxiliary verb, or a clause-modified noun, both annotations are reasonable.

3 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 for the asterisk (*) in *sh*t* works as a mask for 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 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 2 shows the tokenization results, compared to other available tokenizers. The Stanford (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. 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.

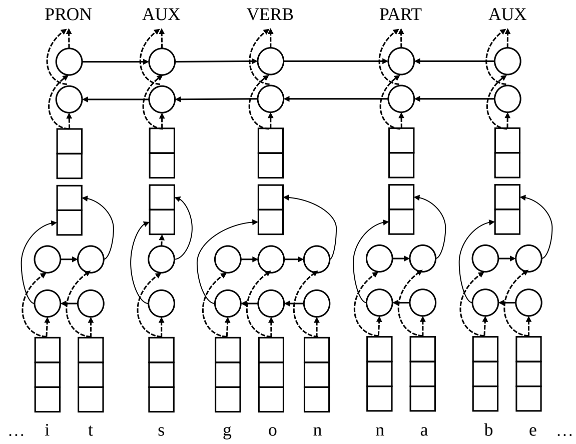

Figure 5: The bi-LSTM POS tagger model that tags 'it s gon na be' as PRON AUX VERB PART AUX.

[i'm a little confused by the pic, need discussion —YI]

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 the twitter specific glove embedding released by Pennington et al. (2014) as the word embeeding in all the neural tagger and parser[9].

[9] 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 3: POS tagger comparison on gold-standard tokens in the TWEEBANK V2 test set.

System	F_1
Stanford CoreNLP	92.0
our tokenizer (§3.1)	93.6

Table 4: Owoputi et al. (2013) POS tagging performance with automatic tokenization on the TWEEBANK V2 test set.

[I wonder if we ought to give more detail about our biLSTM systems, here and under tokenization? will reviewers want to know details? —NAS] [$^{Y}_{L}$I add figure to illustrat our model.]

Experimental results. We tested the POS taggers on TWEEBANK V2 test set. Results with gold-standard tokenization are shown in Table 3. The careful feature engineering in the POS tagger of Owoputi et al. (2013) outperforms our neural 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 4. 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 criterion. We therefore use the neural greedy 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.

Section 2.5 shows that severe ambiguities exist when we create TWEEBANK V2, which makes the training data of the parser very noisy. Frénay and Verleysen (2014) suggested that most of losses are shown to be not robust to classification noise, including the log-likelihood loss used in Ballesteros et al. (2015). To conquer the noisy data and further improve the parsing accuracy, we followed the empirical study of Dietterich (2000) and ensemble of 20 differently initialized parsers. To make it still parses in linear time, we propose a new method to distill the ensemble of 20 parsers into one parser. To our knowledge, this is the first attempt to distill an ensemble of greedy transition-based parsers.

Distilling a simple and fast *student model* from the accurate but slow *teacher model* was explored by Hinton et al. (2015) and Kim and Rush (2016). In their works, the predicted distribution from the teacher model serves as a soft target for training the student model by minimizing

$$\mathcal{L}_{KD} = \sum_k q(y = k \mid x) p(y = k \mid x) \quad (1)$$

where $q(y = k \mid x)$ is the probability distribution predicted by the teacher model. Training the parameters of the student model (p) is accomplished by interpolating between two losses:

$$\mathcal{L} = (1 - \alpha)\mathcal{L}_{NNL} + \alpha\mathcal{L}_{KD} \quad (2)$$

where $\mathcal{L}_{NNL}$ is the negative log-likelihood loss.

Kuncoro et al. (2016) were inspired by the distillation idea and proposed to incorporate the knowledge from an ensemble of 20 greedy transition-based parsers into one single *graph-based* parser by adapting the soft target into a structured loss. However, as Kuncoro et al. (2016) pointed out, it is not straightforward to incorporate the structure loss into a *transition-based* parser. Considering the efficiency advantage of greedy transition-based parsers over their graph-based counterparts, it is worth considering distillation of an ensemble into a greedy parser.

The generic algorithm for training a transition-based greedy parser usually includes 1) generating a sequence of transitions from the training data, called the "oracle" and 2) learning the classifier from the oracle transitions. One natural adaptation of the distillation will be interpolating $\mathcal{L}_{NLL}$ with the distillation objective (Equation 2) in the second step. We name this method DISTILL EXPERT.

Preliminary results. In our preliminary experiments, we train our parser on the combination

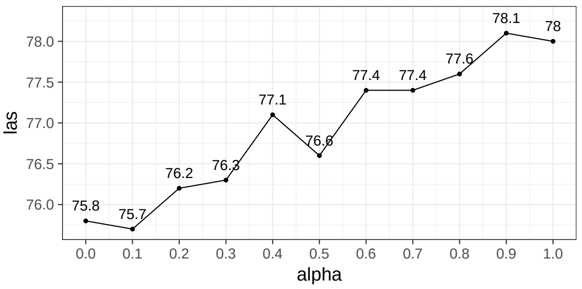

Figure 6: The effect of α on EXPERT DISTILLATION.

System	UAS	LAS
Kong et al. (2014)	81.6	77.2
Dozat and Manning (2016)	81.9	77.7
Ballesteros et al. (2016)	80.4	75.8
Ensemble (20)	83.5	79.4
DISTILL EXPERT (α=1.0)	82.2	78.0
DISTILL EXPERT (α=0.9)	82.4	78.1
DISTILL EXPLORE	82.5	78.4

Table 5: Dependency parser comparison on TWEEBANK V2, with automatic POS tags.

System	UAS	LAS
Kuncoro et al. (2016)	94.3	92.1
Dozat and Manning (2016)	95.7	94.1
Dyer et al. (2015)	93.0	90.9
Ensemble (20)	94.5	92.6
DISTILL EXPERT (α=1.0)	93.8	91.7
DISTILL EXPLORE	94.1	92.0

Table 6: Dependency parser comparison on PTB test set.

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, including the value of α in Equation 2, are tuned on the TWEEBANK V2 development set. Unlabeled attachment score and labeled attachment score including punctuations are used to evaluate parsing performance.

Reimers and Gurevych (2017) pointed out that the training process for neural networks are highly non-deterministic and largely depends on the seed value for pseduo-random number generator. Our preliminary experiments confirm this finding by witnessing a gap of 1 LAS on development set between the best (75.8) and worst (74.8) runs. To eliminate the effect of random number, for each hyperparameter setting, we conduct 5 runs with different seeds and report the averaged score.

Our 20 parsers are ensembled by averaging their output probability distributions over transitions. The ensemble parser achieves a LAS of 79.4, which is more than 3 points higher than the baseline single parser. Distilling from this parser leads to a single parser with 78.1 LAS.

Effect of α. We further study the effect of the interpolation hyperparameter α by varying it from 0 to 1. The results are shown in Figure 6. We can see that distilling parsers with larger α achieves better performance, which means the model should pay more attention to learning from the teacher model. Indeed, the negligible difference between the best performance at $\alpha = 0.9$ and $\alpha = 1.0$ casts doubt on the necessity of including $\mathcal{L}_{NNL}$ in the training objective at all.

Learning from exploration. We further set α to 1.0, eliminating the oracle from the usual transition-based parser training algorithm. Instead of generating training instances from the oracle, we generate them by random sampling transitions from the teacher (ensemble) model's output distribution and follow the trajectory of these transitions. We name this method DISTILL EXPLORE.

Final results. Our final parsing experiments are shown in Table 5. We also compare our model with the TWEEBOPARSER of Kong et al. (2014) and the state-of-the-art parser (Dozat and Manning, 2016) in CoNLL 2017. Both the compared systems are re-trained on the same data as our system. From Table 5, our baseline system is 1.9 point behind the state-of-the-art parsing systems. However, through distilling, our baseline parser catched up and outperform that of Dozat and Manning (2016) and DISTILL EXPLORE further outperforms DISTILL EXPERT, gaining 0.4 point in LAS improvement. We attribute the improvements of DISTILL EXPLORE over DISTILL EXPERT to the former's exploration of ensemble-sanctioned parser states of different qualities.

We also perform additional experiments on the Penn Treebank were performed following the setting of Dyer et al. (2015); Table 6 shows a similar trend, and we achieve performance close to that of the graph-based parser of Kuncoro et al. (2016). However, our model is still way behind Dozat and Manning (2016). We address this to the

Pipeline	F-score
Tokenization	98.3
POS tagging	93.7
Dependencies	74.1

Table 7: Evaluating our model in pipeline manner.

intrastic differences between transition-based and graph-based parsing algorithm.

3.4 Final Evaluation

Finally, we evaluate the problem of parsing tweets into UD in a pipeline manner. The results are shown in Table 7. From this result, we can see that in the whole pipeline, every component plays an important role. Without gold segmentation, the parsing performance drops more than 4 points.

4 Conclusion

In this paper, we study the problem of parsing tweets into universal dependencies. We adopt 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 annotations and show that it is challenging to deliver consistent annotation due to the ambiguities in ways of understanding and explaining the tweets. On this new treebank, we build a pipeline system to parse tweets into UD. We propose a new method to distill an ensemble of 20 parsers into a single one to conquer the annotation noise without sacrificing efficiency. Our parser achieves an improvement of 2.3 in LAS over the baseline and outperforms other state-of-the-art parsers.

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.

Chris Dyer, Miguel Ballesteros, Wang Ling, Austin Matthews, and Noah A. Smith. 2015. Transition-based dependency parsing with stack long short-term memory. In *Proc. of ACL*.

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

Tim Finin, William Murnane, Anand Karandikar, Nicholas Keller, Justin Martineau, and Mark Dredze. 2010. Annotating named entities in twitter data with crowdsourcing. In *Proc. of NAACL HLT Workshop on Creating Speech and Language Data with Amazon's Mechanical Turk*.

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.

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