Parsing Tweets into Universal Dependencies

Anonymous ACL submission

Abstract

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 tweets treebank (TWEEBANK) which has 12,149 tokens and was created using the Yamada and Matsumoto (2003) dependencies. Both these 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 tweets treebank of 55,607 tokens by following the UD guidelines but also contending the domain-specific challenges which wasn't covered by UD guidelines. Our annotation includes a whole pipeline of tokenization, part-of-speech (POS) tags and universal dependencies.

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 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 an improvement of more than 3 points in LAS over baseline parser can be achieved with our distillation method.

Contribution of this paper includes:

- 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 dependencies annotation.

- We built up a whole pipeline to parse the raw tweet text into universal dependencies (§3). We propose a new distillation method in training the parser in our pipeline which achieve 3 points improvement without sacrificing efficiency.

We release our system at.

[not clear to me: did we reannotate the POS corpus? was that automatic or manual? I wasn't sure what to do with the POS section below. —NAS] [Y/L we re-annotated the POS. In one way, Gimple's POS is not very consistent with UD POS. In another way, 2,500 new tweets were annotated from scratch and they didn't have Gimple's tags.]

2 Annotation

[need to introduce this section and explain the structure. I'm not crazy about what I have now, but I like it better than what we had before, which seemed to keep repeating the same ideas in different contexts. the flow still seems wrong; we start by talking about dependencies, then take a detour to POS, then come back to dependencies. —NAS]

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 a layer to a portion of the data annotated with part-of-speech tags by Gimpel et al. (2011) and Owoputi et al. (2013). They also contributed a system for parsing; we defer discussion of their parser to §3.

Kong et al.'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 Part-of-Speech Annotation

Before turning to UD annotations, we (re)annotated the data with part-of-speech, 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 (UD_English) [1] [add cite —NAS] 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 part-of-speech 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 O'Connor et al. (2010). In particular, they opted out of tokenizing contractions and possessives, introducing new parts of speech instead.[2] For us, these tokens must be split, but universal parts of speech can be applied.

2.3 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

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

[2] These tags only accounted 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	0.49
sum	2.08	7.55

Table 1: Proportion of non-syntactic tokens in the annotation.

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") 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. [explain what we do, and include how this is similar to or different from Kong et al —NAS][$^{Y}_{L}$ done.]

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 is in the same spirit with his work. Further study shows that the abbreviations which clearly carry syntactic functions only make 0.06% tokens in our dataset, which also indicates it's unnecessary to normalize text in social media. [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 functions. In our annotation, there are five types of non-syntactic tokens which include sentiment emoticons, retweet markers, topical hashtags, referential URLs, and truncated words. Figure 1 illustrates examples of these non-syntactic tokens. In the sense of part of speech, emoticons are tagged as *symbol* (SYM) by following the UD guidelines. For other 4 types of non-syntactic tokens, they generally have the *other* (X) part of speech. In our annotation, these five types of tokens make more than 7.5% of all tokens and detailed statistics can be found in Table 1. [in latex source I've commented out a paragraph about truncated content. it's not clear to me what the nonsyntactic tokens are, when that happens —NAS][$^{Y}_{L}$ i've put truncated words here.]

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.

[I'm confused about (1) how we define non-syntactic tokens (what are the criteria) and (2) whether this is something we kick over to POS annotation —NAS] [$^{Y}_{L}$ I've put the definition of non-syntactic tokens there.]

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.

[explain our conventions and how they relate to Kong and to UD. in the draft before, we said that we had conventions, and Kong didn't, but we didn't really say what ours were. —NAS] [$^{Y}_{L}$ The bullet points describe our convention for non-syntactic tokens.]

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 help us avoid confusion. [say what it is, and what they attach to, and clarify that the annotators didn't do this manually (I think) —NAS]

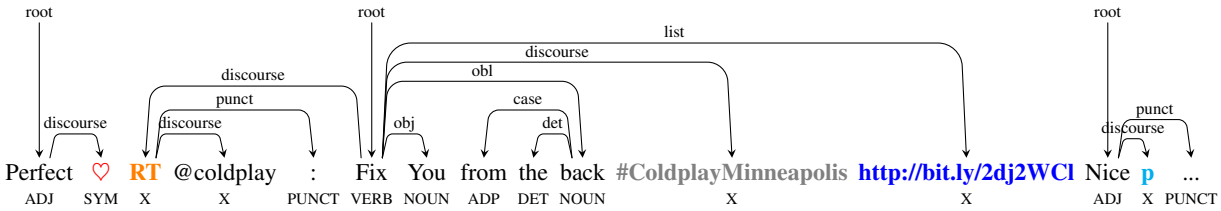

Figure 1: A example tweet contains major non-syntactic tokens, with sentiment emoticon, retweet marker, topical hashtag, referential URL, and truncated word.

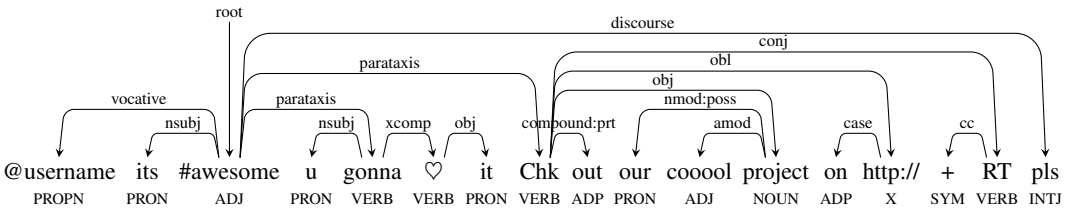

Figure 2: An example tweet with informal but syntactic tokens.

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). [need an example. —NAS] [YL done.]

2.4 TWEEBANK V2

Following the guideline mentioned above, we create a new annotated Twitter treebank, which we call TWEEBANK V2.

2.4.1 Data Collection

TWEEBANK V2 is built on the original data of TWEEBANK V1 (Kong et al., 2014, 639 tweets), along with additional 201 tweets sampled from Gimpel et al. (2011) and 2,500 tweets sampled from the Twitter stream from Feb. 2017 to Jul. 2017. The latter data source consists 147.4M English tweets which are filtered by *lang* attribute in the tweet JSON and *langid.py*[3] toolkit.

2.4.2 Annotation Process

The whole annotation process contains three stages. In the first stage, 18 researchers worked on the TWEEBANK V1 proportion and created the initial annotation in one day following our guideline. Both the guideline and annotation were further refined by the authors of this paper to increase the coverage of our guideline and solve inconsistency between different annotators. In the second stage, a postagger and parser are trained on the annotated data from the first stage, and automatically analyze the sampled 2,500 tweets and the authors of this paper manually correct the parsed data. Finally, an extra layer of word-level normalization was manually annotated.

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

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

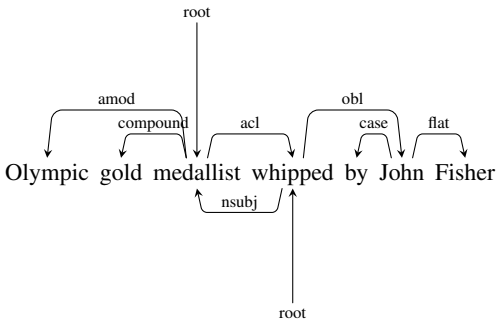

Figure 3: An example of disagreement when annotating tweets with ellipsis.

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. The disagreements on dependencies annotation come in several different ways. One of these disagreements is resulted by the ellipsis as shown in Figure 3. Due to the absence of auxiliary verb, this tweet can either be treated as a nominal phrase modified by a clause or a sentence with auxiliary verb missing.

3 Pipeline

We present a pipeline system to parse tweets into universal dependency.

3.1 Tokenizer

Tokenization, as the initial step of many NLP tasks, is non-trivial for Twitter domain (O'Connor et al., 2010). Due to the informal nature, tweets present several cases which are unconventional for traditional toeknization module, like hashtags, atmentions and emoticons. Some of them are even ambiguous without understanding their contexts. For example, the asterisk (*) in *4*3* works as a symbol of multiplying and delimiter of tokens *4 * 3*; but for the asterisk (*) in *SH*T* works as a mask for censorship, should not be treated as a delimiter.

Considering the complexity of tweets tokenization, instead of manually crafting rules, we propose to deal with tweets tokenization with a character-level bidirectional LSTM (bi-LSTM) sequence-labeling model. Our model takes the raw sentence and tag each character in this sentence as whether it is the beginning of a word.

System	F
Stanford Tokenizer	96.6
Twokenizer	94.4
UD pipe v1.2	97.3
Our bi-LSTM tokenizer	**98.3**

Table 2: The tokenization results.

System	P
Stanford Postagger	90.4
ARK tagger (greedy)	94.2
ARK Tagger (CRF)	**95.1**
Ma and Hovy (2016)	
Our word bi-LSTM	89.1
Our char bi-LSTM + word bi-LSTM	91.4

Table 3: The POS tagging result on gold tokenization.

Experimental Results We test our tokenizer on Tweebank v2 test set. F-score was used to evaluate the tokenization accuracy. Our model was trained on a merge of UD_English and Tweebank v2 training data. Table 2 shows the tokenization results along with comparison with other tokenizers. UD pipe v1.2 (Straka and Straková, 2017) model was re-trained on the same data. Our bi-LSTM tokenizer achieves the best accuracy among all these tokenizers. From this table, we can also see that the model-based tokenizers generally outperform the rule-based tokenizers, which shows the necessity of using statistical model for tweet tokenization.

3.2 POS-Tagger

Part-of-Speech tagging for tweets has been studied in previous literals (Ritter et al., 2011; Gimpel et al., 2011; Owoputi et al., 2013; Gui et al., 2017). The goal is to give a fair comparison of these models on our annotation. We are especially interested in finding out the can neural network models win against the model enhanced by carefully crafted features.

Experimental Results We test the POS taggers on Tweebank v2 test set and the results are shown in Table 3. All the comparison systems were retrained on the merge of UD_English and Tweebank v2 training set. Gold tokenization was used. This table shows that carefully feature engineer POS tagger (Owoputi et al., 2013) outperform the neural network models.

We also study the influence of tokenization on POS tagging by evaluating the model's performance on automatically tokenized data. We use

System	F
Stanford Tokenizer	92.0
Our Tokenizer	93.6

Table 4: The POS tagging result on automatic tokenization.

the best performed POS tagger (Owoputi et al., 2013) on gold tokenization as our POS tagger. The results are shown in Table 4. Table 4 shows that tokenization errors propagate to POS tagging but these errors didn't significant influence the tagging decisions.

3.3 Parser

Social media applications usually require processing tones of data. Speed becomes important under such circumstance. In this paper, we use the neural greedy parser by Ballesteros et al. (2016), which can parse a sentence in linear time and harness the useful character representations. To further improve the parsing accuracy, we propose a new method to distill an ensemble of 20 differently initialized parsers into one parser.

Distilling a simple and fast *student model* from the accurate but slow *teacher model* was explored in 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

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

where $q(y = k|x)$ is the probability distribution predicted by the teacher model. Training the student model is delivered by training with a interpolate between two losses:

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

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

Kuncoro et al. (2016) was inspired by the distillation idea and proposed to incorporate the knowledge from an ensemble of 20 parsers into one single graph-based parser by adapting the soft target into a structure loss. However, as Kuncoro et al. (2016) pointed out, it's not straight-forward to incorporate the structure loss into a transition-based parser. Considering the efficiency advantage of transition-based parser over its graph-based counterpart, it's worth study distillation of an ensemble of different parsers.

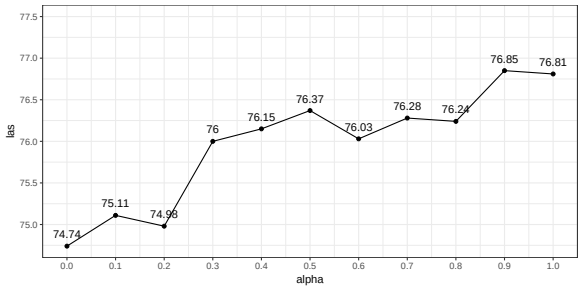

Figure 4: The effect of α in distillation.

The generic algorithm for training a transition-based greedy parser usually includes 1) generating a sequence of transitions from the training data with "oracle" and 2) learning the classifier from the transitions. One natural adoption of the distillation will be replacing $\mathcal{L}_{NLL}$ with the distillation objective (Equation 2) in the second step. We name such method as *distill expert*. We adopt such distilling method in our preliminary experiments.

Preliminary Results In our preliminary experiments, we train our parser on the merge of UD_English and Tweebank v2 training set. Automatic POS tags are assigned with 5-fold jackknifing. Hyper-parameters including the value of α in Equation 2 are tuned on Tweebank v2 development set. The 20 parsers are trained with different random seeds and ensembled by averaging their output probability distribution. The ensemble parser achieves a LAS of 78.8, which is more than 3 points higher than the baseline single parser. Distilling from this parser leads to a single parser of 77.0 LAS.

Effect of α We further study the effect of α by varying it from 0 to 1 and the results are shown in Figure 4. From this figure, we can see that distilling parsers with larger α achieves better performance, which means the model should pay more attention to learn from the teacher model. And the changes of α from 0.9 to 1.0 didn't significantly influence the final result, which casts a doubt on the necessity of learning from $\mathcal{L}_{NNL}$.

Learning from Exploration We further set α to 1.0 and this enable us to get rid of the "oracle" in the first step in generic transition-based parser training algorithm. Instead of generating training instances from the oracle, we generate them by random sampling transition from the teacher model's output distribution and follow the trajectory of these transitions. We name such method as *distill*

System	UAS	LAS
Ballesteros et al. (2016)	80.2	75.4
Ensemble 20x	83.0	78.8
Distill 20x expert (α=1.0)	81.2	76.7
Distill 20x expert (α=0.9)	81.4	77.0
Distill 20x explore	82.2	77.9

Table 5: The parsing results on automatic POS tags.

System	UAS	LAS
Dyer et al. (2015)	93.0	90.9
Ensemble 20x	94.5	92.6
Distill 20x expert (α=1.0)	93.8	91.7
Distill 20x explore	94.1	92.0
Kuncoro et al. (2016)	94.3	92.1

Table 6: Distillation on PTB.

explore.

Final Results Our final parsing experiments are shown in Table 5, in which we can see that distilling from exploration significant outperform distilling from expert by achieving one more point LAS improvement. Additional experiments on the Penn Treebank data are performed following the setting of Dyer et al. (2015) and similar trend are witnessed. We address this improvement to the fact that learning from exploration enables the single parser to learn from the ensemble on states of different qualities.

3.4 Final Evaluation

Finally, we evaluate the tweet parsing in a pipeline manner.

4 Related Work

5 Conclusion

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*.

Danqi Chen and Christopher Manning. 2014. A fast and accurate dependency parser using neural networks. In *Proc. of EMNLP-2014*.

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.

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.

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*.

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.

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.

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

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.

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*.