Parsing Tweets into Universal Dependencies

First Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
`email@domain`

Second Author
Affiliation / Address line 1
Affiliation / Address line 2
Affiliation / Address line 3
`email@domain`

Abstract

1 Introduction

Analyzing the syntax of tweets is challenging for traditional NLP tools because most of the tweet texts are informal and noisy.

In this paper, we propose to parse the tweets in the convention of universal dependencies and built up the whole pipeline to parse the tweets from the raw text form.

Contribution of this paper includes:

- We create a new version of tweet Treebank Tweebank 2.0
- We propose a neural network method to parse tweets into universal dependencies
- We study the adaptation of universal dependencies for analyzing tweets

2 Data

2.1 Linguistic Phenomena of Twitter

Twitter, as an extreme example of informal domain, contains a collection of conversational languages (like abbreviation, informal contraction, and variant entity names) and twitter-specified markers (like retweet mark, and username). Despite the informal nature of tweets, we argue that they basically follow the same grammar as their formal language counterpart. The key issue lies in understanding the specific construction that doesn't carry any syntactic functions. In the following section, we studied such constructions from both the token and structure level. [$^{Y}_{L}$I DON'T LIKE THE NAME OF **STRUCTURE LEVEL**]

2.1.1 Token Level

Informal Syntactic Tokens. Informal tokens like phrase abbreviation (like `mfw`: my face when, `rn`: right now, and etc.) and orthographic variants (like `sooo cooool`) have drawn much attention in previous literals (Finin et al., 2010; Eisenstein, 2013) [$^{Y}_{L}$NOT SURE ORTHOGRAPHIC VARIANTS IS A GOOD NAME, NEED REFERENCE.] However, in the sense of syntactic functions, they work in the same way as the ordinary words, which makes them requires less specialty in designing annotation conventions.

Non-syntactic Tokens. Besides the informal but syntactic tokens, there are a large collection of tokens that cannot be handled by the syntactic theory designed for standard text. A major part of these tokens consists of meta data in tweets, like the sentiment emotion, retweet mark, topical hashtag, and referential URL shown in Figure **??**.

However, whether a token convey some syntactic function cannot be simply decided by its form. For example, besides being a discourse marker, `RT` can also works as a abbreviation of the verb `retweet`. Emoticons are sometimes used as a nominal part of the sentence like 'I ♡ it.' Twitter users have developed a casual habit of using hashtag, even with words expressing exclamation (like `#lol`). Thus, whether a token make a non-syntactic one is highly

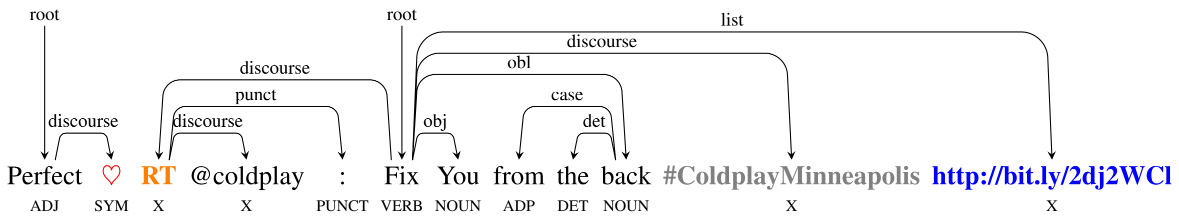
root
root
list
discourse
obl
discourse
punct
case
discourse
discourse
obj
det
Perfect ♡ RT @coldplay : Fix You from the back #ColdplayMinneapolis http://bit.ly/2dj2WCl
ADJ SYM X X PUNCT VERB NOUN ADP DET NOUN X X

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

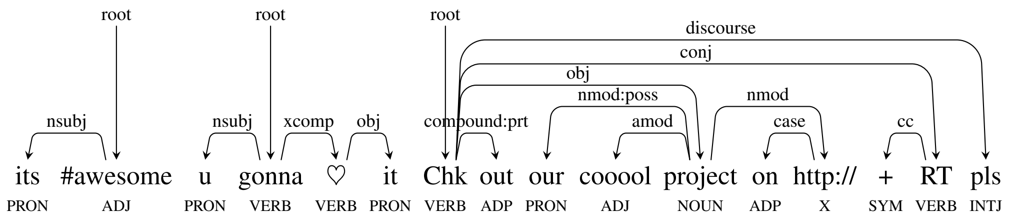
root
root
root
discourse
conj
obj
nmod:poss
nmod
nsubj
nsubj
xcomp
obj
compound:prt
amod
case
cc
its #awesome u gonna ♡ it Chk out our cooool project on http:// + RT pls
PRON ADJ PRON VERB VERB PRON VERB ADP PRON ADJ NOUN ADP X SYM VERB INTJ

Figure 2:

conditioned on its context.

A kind of token that doesn't carry clearly syntactic functions can be resulted by the 140 characters limits of tweets. Since last tokens that exceeding 140 characters limits are truncated in tweets which leaves a broken sentence, it is impossible to fully recover their original form and figure out their syntactics. In this paper, we propose to treat these truncated tokens, along with the other 4 kinds of tokens mentioned above, as ones without syntactic functions.

In the sense of dealing with the non-syntactic tokens, Kong et al. (2014) proposed an preprocessing step named *token selection* to remove non-syntactic tokens from the original tweets. However, concrete standard of non-syntactic tokens was not presented in their work. What's more, these unselected tokens are absent from downstream processing, including parsing. We differ from their work by proposing concrete standard of non-syntactic tokens and we include these non-syntactic tokens in our final analyzed dependencies by adding special attachment to them.

2.1.2 Structure Level

Besides the token level phenomena of twitter, there are also structural patterns appearing in the twitter.

- Retweet structure: `RT @user : {tweet content}` is a typical structural pattern when a user is retweeting from other users.
- At-mentioned structure: Except the retweet structure and syntactically at-mentioned username, we think that all of other at-mentions are vocative case, usually appearing at the beginning or the end of the sentences in tweets.
- Paratactic parts without punctuations: We use "part" to refer to a self-contained clause or phrase, which is independent of other "part". Very often, one tweet is comprised of multiple parts without any delimiting punctuations, such as `{part 1} {part 2} ...`

We treat retweet structure and at-mentioned structure as structural patterns and keep their annotations consistent across tweets. For parataxis case, we treat each single sentence or phrase normally, and connect them together following the UD conventions.

From Fig. **??**, we argue that `u` and `rn` are syntactic tokens. `th` is a truncated token and does not have any syntactic function, although we can infer that the `th` could be probably `the`. `#ACL2017`, `:)` and `http://url` are optionally syntactic tokens, and should be all treated as non-syntactic tokens in this example. `RT @Yijia :` should be analyzed as the retweet structure, and `heading to`

`Canada for ACL` and `can u gimme some money` should be considered as paratactic clauses without punctuations.

However, consider another example in Figure 2. All of the optionally syntactic tokens (`RT`, `http://url`, `#NAACL`, `#Awesome`, `@zypandora`, `@nlpnoah` and ♡) have syntactic functions, and we need to take them into account in the syntactic analysis.

2.2 POS Tagging

We follow the UD V2 morphology guideline and use the Universal POS tags (**?**) to tag the tweet tokens. For twitter specific linguistic phenomena, we have set different strategies and try to align with the UD conventions and UD_English as close as possible.[1]

In token level phenomena, for non-syntactic tokens, we tag most of them as X (other), except for the emojis and emoticons, which would be tagged as SYM, as same in the UD_English. For syntactic tokens, we tag them considering their corresponding syntactic function.

In structure level phenomena, we think `RT @user` in retweet structure have no contribution to the tweet syntax and tag both of them as X. We tag usernames in at-mentioned structure as PROPN as they present the vocative case.

In Fig. **??**, `RT @yijialiu`, `#ACL2017`, `http://url` and `th` are all tagged as X and `:)` is tagged as SYM. `rn` is tagged as ADV. We should note that multiple tokens are actually included in a phrasal abbreviation, and we use one of their tags as the tag of the whole abbreviation according to their dependency relation or semantic priority. If there is a hierarchy in the abbreviation, we use the tag of the head word. In UD, “Right” is the dependent of “now ” in the dependency of `rn`, so we tag `rn` as ADV. In Fig. **??**, `RT` and ♡ would be tagged as VERB, `http://url` as X, `#Awesome` as ADJ, `#NAACL`, `@zypandora` and `@nlpnoah` as PROPN. One exception is that we tag URLs always as X regardless whether they are syntactic tokens. The reason is that we observed, in UD_English, that the URLs are all tagged as X and we want to conform with the UD dataset, although we think it is not appropriate and should be changed to tags such as PROPN when they are treated as syntactic tokens.

?) proposed a twitter POS tagset that handles most of the token level phenomena. However, we found that their methods did not conform with UD conventions and our strategies have some advantages over their tagset. We discuss the main difference as follows. The first difference lies in the tokenizations. **?**) opted not to split contractions or possessives and introduced four new tags (S, Z, L, M) for combined forms: {nominal, proper noun} × {verb, possessive}. Major concern of designing such tags is to minimize the effort of tokenization, but it is not comprehensive and far from including all of the possible combinations of POS tags within the contractions. What’s more, only a small proportion of tokens can be categorized into these tags (2.7 % in total), which casts a doubt of the usefulness of these tags. Instead UD conventions suggest that we put such complexity into the tokenization module and split contractions and possessives, then tag them accordingly, so that we do not need extra POS tags for the combined forms. The second difference is that although the phrasal abbreviations are not split in both methods, they use tag G for all the phrasal abbreviations, while we use the tag of the head word or the highest hierarchical word in terms of dependencies as the tag of the whole phrasal abbreviations. We think it is not reasonable to treat the phrasal abbreviations in the same part of speech because obviously abbreviations can have different syntactic functions and we want to preserve the most useful information for both parser and the downstream tasks. For example, `wtf` could be tagged as PRON or INTJ according to the context, and `rn` is usually tagged as ADV. Tagging them merely as G will definitely lose much information. Third, special tags were designed to handle twitter or online special tokens such as URLs, hashtags and emoticon in **?**). However, in most cases, as long as theses tokens are non syntactic, they play the same role in tweets, hence should be tagged the same. In this paper, except for the emoticon and emoji, tagged as SYM, we tag all the other non-syntactic tokens as X.

2.3 Dependency Annotation

We follow the UD V2 syntax guideline and the English specific Universal Dependency Relations to

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

annotate our data.

In token level phenomena, we adopt the following rules for non-syntactic tokens, denoted as x:

- If the sentence x belongs to has a clear predicate with syntactic function, x should not have any dependent and is attached to this sentences predicate;
- If the sentence x belongs to consists of only non-syntactic tokens we attach x to the first token of the sentence;
- In parataxis case, when it is difficult to decide which sentence or phrase x should belong to, we always attach x to the previous predicate of the paratactic clause or phrase;
- x is mostly labelled as *discourse*, but URLs are always labelled as *list*;

The reason we annotate URLs differently is the same as in POS tagging, that UD_English labelled them as *list* when they are non-syntactic tokens and we want to keep the dataset conformed with it.

For syntactic tokens, they are considered the same as normal tokens and we just annotate them by following the UD guidelines.

we further adopt the following rules for structure level phenomena:

- Retweet structure: we attach `@user` to `RT` and `RT` to the predicate of the following sentence with *discourse* as labels.
- At-mentioned structure: we attach at-mentioned username to the predicate of the most relevant sentence with *vocative* label. If it is hard to decide which sentence would be the most relevant, we attach it to the previous sentence's predicate.
- Paratactic parts without punctuations: we follow the UD convention, set the first part as the main part of the sentence, and set its predicate as the main predicate of the sentence. Every predicate of the following parts should attach to the main predicate. If the main part and some following part are both phrases, the label is *list*, otherwise the label is *parataxis*. Note that we still have an exception for URLs, conforming with UD, which are always labelled as *list*, regardless of the type of the other part.

Kong et al. (2014) built a syntactic treebank of tweets called Tweebank using Gimpel et al.'s POS tagset. Apart from the difference in POS tagging, we found the following difference in both methods. First, Tweebank is an unlabelled dependency treebank. There is clearly a doubt about the usefulness of the Tweebank. We annotate the labelled dependencies by following the UD annotation guideline. Second, most of Tweebank was built in a day by two dozen annotators, most of whom had only cursory training in the annotation scheme (Kong et al., 2014). We think, after investigation, that despite the speed, the quality of data is still to be improved. We follow strictly the UD annotation guideline and only make twitter specific conventions when there is no suitable convention to adopt in UD guideline or no relevant data found in UD_English. Third, Kong et al. (2014) developed a first-order sequence model to filter out the non-syntactic tokens and punctuations, and exclude them before parsing. It is very different from most of the annotation conventions, especially for UD conventions. We believe that every token should be included in the dependency tree, whether they have syntactic functions or are to be evaluated in the end, as in most of the conventions, so we annotate all tokens. Last, Kong et al. (2014) adopt the "Yamada-Matsumoto" conventions (**?**), where auxiliary verbs are parents of main verbs, and prepositions are parents of their arguments, in contrast to UD conventions, where content words are put into the primacy, and dependency relations hold primarily between them, rather than being indirect relations mediated by function words. Auxiliary verbs and prepositions are all function words, and are attached to the most related content words in UD.

2.4 Tweebank V2

We present Tweebank V2, a version 2 of Tweebank (Kong et al., 2014).

The tweet source of Tweebank V2 consists of three parts. The first is the original Tweebank created by (Kong et al., 2014), denoted as Tweebank V1. There are 639 unique tweets in the training data of Tweebank V1, and 201 unique tweets in the data set. All of the unique tweets in Tweebank V1 are included in Tweebank V2. The second are 210 tweets from POS-tagged Twitter corpus of **?**), denoted as Tweebank V1 Dev. It was initially created as a development set of Tweebank V1 for hyperparameter tuning. Tweebank V1 Dev comprises were equally sampled from two data set, OCT27 (tweets sampled from a particular day, October 27, 2010) and DAILY 547 (one random English tweet per day from January 2011 through June 2012, 547 tweets in total). We should note that all of the Tweebank V1 tweets were also drawn from OCT27 and DAILY547, and the new tweets we extracted are not in the Tweebank V1. The third are 2500 tweets from the Twitter streams from February 2017 to July 2017 from ArchiveTeam,[2] which we call Feb_Jul_16 corpus. The Twitter streams are in Spritzer version, which provides a 1% random sample of public tweets everyday, and the tweets are in twitter JSON format. There are xxx tweets in all the streams. We filtered them with *lang* attribute in *user* and used *langid.py*[3] to pick out English tweets, which ends in xxx tweets in English. Then we sampled 2500 tweets roughly equally from these months.

2.4.1 Basic Statistics

Tweebank V2 contains 3550 tweets in total, split into training set with 1639 tweets, development set with 710 tweets and test set with 1201 tweets. The training set contains all the 639 unique tweets in the training set of Tweebank V1 and another 1000 tweets in feb_jul_16 corpus. The development set contains 210 tweets of Tweebank V1 Dev and another 500 tweets in feb_jul_16 corpus. The test set contains all the 201 tweets in the test set of Tweebank V1 and another 1000 tweets in feb_jul_16 corpus.

We process all the raw tweets of these corpora from scratch, from tokenization and POS tagging to dependency annotation. As Tweebank V1 and the POS-tagged Twitter corpus of **?**) used a tweet tokenizer (**?**) and POS tagger (**?**) that do not conform with UD guidelines, as we discussed before, we did not use any data of the intermediate steps from these corpora. The basic statistics of Tweebank V2 is shown in Table 1.

[2]https://archive.org

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

	Train	Dev	Test
tweets	1639	709	1201
tokens	24753	11742	19112
types	7579	4165	6228
parts	3025	1402	2252

Table 1: Basic Statistics of Tweebank V2

2.4.2 Twitter Specific Statistics

Following the discussion of the Twitter specific linguistic phenomena, we obtain Table 2.

3 Pipeline

4 Model

5 Experiments

6 Related Work

Eisenstein (2013) reviewed NLP approaches for analyzing text on social media, especially for tweets and showed that there are two major directions for NLP community to handle the tweets, including normalization and domain adaptation. He also pointed out that normalization can be problematic because precisely defining the normalization task is difficult.

Kong et al. (2014) argues that the Penn Treebank approach to annotation is poorly suited to more informal genres of text, as some of the annotation challenges for tweets, including token selection, multiword expressions, multiple roots, and structure within noun phrases diverge significantly from conventional approaches. They believe that rapid, small scale annotation efforts performed by imperfectly-trained annotators should provide enough evidence to train an effective parser, given the rapidly changing nature of tweets (Eisenstein, 2013), the attested difficulties of domain adaptation for parsing (**?**), and the expense of creating Penn Treebank-style annotations (**?**). Therefore, they build a new corpus of tweets (Tweebank), with conventions informed by the domain, using new syntactic annotations that can tackle all the forementioned problems annotated in a day by two dozen annotators, most of whom had only cursory training in the annotation scheme. Then, they modify the decoder of the TurboParser, a graph-based dependency parser, which is open-source and has been found to perform well on a

				Train	Test	Dev	All	Per
token	syntactic	phrasal abbreviation						
	optionally syntactic	RT	syntactic					
			non-syntactic					
		hashtag	syntactic					
			non-syntactic					
		at-mentioned	syntactic					
			non-syntactic					
		emoticon/emoji	syntactic					
			non-syntactic					
	non-syntactic	truncated						
structure	retweet							
	at-mentioned							
	parataxis without punctuations							

Table 2: Twitter Specific Statistics of Tweebank V2

range of parsing problems in different languages (**?**) to adapt to the Tweebank dataset, and incorporate new features such as Brown Clusters and Penn Treebank features and changes to specification in the output space into TurboParser.

Like "mfw" is usually followed by an adverbial clause and "ima" is usually followed by a clausal complement. It is not reasonable to treat them in the same part-of-speech. In this paper, when annotating the POS tagging for abbreviations, we first try to recover their original forms, then use the POS of the core-word as the POS for the abbreviation. Second, four special POS tags (S, L, M, Y) were designed to handle contraction words in Gimpel et al. (**?**). Major concern of designing such tags is to minimize the effort of tokenization. However, contractions of common nouns and pronouns are casted into the same category which increase the difficulty of distinguishing their syntactic function (say, there's and book'll are treated with the same syntactic function). What's more, only a small proportion of words can be categorized into these tags (2.7 % in total), which cast a doubt of the usefulness of these certain tags. In this paper, we believe such contraction can be properly handled by tokenization module, so we suggest to tokenize the contraction word and annotate POS tag accordingly. Besides the contraction that be conventionally tokenized, tweets also witness a set of unconventional contraction like iv (I've), whatis (what is). In this paper, we follow the same idea of annotation abbreviation to handle the unconventional contractions and use the POS of core word of the original form as their POS. Third, special POS was designed to handle emoticon in Gimpel et al. (**?**). However, in most cases, emoticon plays the same role as most of the symbolic tokens. In this paper, we follow the UD guideline to annotate the emoticon as symbol (SYM). At last, its arguable that some of the hashtags, URLs can work as a nominal in tweets. Whether treating them as the same part-of-speech or different ones according to their context is an open question. A preliminary survey on the standard UD English data shows that URL, email address are all tagged as the foreign language (X), so we also tag them as X and leave the disambiguation of their syntactic function to the annotation of parse tree.

7 Conclusion

References

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

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*.

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, October.