#hardtoparse: POS Tagging and Parsing the Twitterverse

Jennifer Foster[1], **Özlem Çetinoğlu**[1], **Joachim Wagner**[1], **Joseph Le Roux**[2]
Stephen Hogan[1], **Joakim Nivre**[3], **Deirdre Hogan**[1] and **Josef van Genabith**[1]
[1]National Centre for Language Technology, Dublin City University, Ireland
[2]LIF - CNRS UMR 6166, Université Aix-Marseille, France
[3]Department of Linguistics and Philology, Uppsala University, Sweden
[1]{jfoster,ocetinoglu,jwagner,shogan,dhogan,josef}@computing.dcu.ie
[2]joseph.le-roux@lif.univ-mrs.fr
[3]joakim.nivre@lingfil.uu.se

Abstract

We evaluate the statistical dependency parser, Malt, on a new dataset of sentences taken from tweets. We use a version of Malt which is trained on gold standard phrase structure Wall Street Journal (WSJ) trees converted to Stanford labelled dependencies. We observe a drastic drop in performance moving from our in-domain WSJ test set to the new Twitter dataset, much of which has to do with the propagation of part-of-speech tagging errors. Retraining Malt on dependency trees produced by a state-of-the-art phrase structure parser, which has itself been self-trained on web material, results in a significant improvement. We analyse this improvement by examining in detail the effect of the retraining on individual dependency types.

Introduction

While much progress has been made on supervised approaches to common natural language processing tasks such as part-of-speech tagging and syntactic parsing, many obstacles still remain before these problems can be said to be solved. The problem of domain adaptation is a well known one within the NLP and the machine learning community. How can a tool trained on one linguistic genre be adapted to another without access to substantial amounts of labelled data? The challenge becomes yet more daunting when we face, not just a new target domain, but the rapidly evolving, linguistically diverse mix of domains that is Web 2.0. In this paper, we examine the problem of adapting a pipeline dependency parsing system, trained on edited newswire, to the language of Twitter.

A dependency-based representation of syntactic structure is appealing because it captures people's notions of grammatical relations more intuitively than phrase structure, because it is a natural mode of representation for languages with a free word order and because parsing algorithms exist, which, when combined with enough training data and an adequate probability model, can produce dependency trees reasonably accurately in linear time. *Labelled* dependency representations are particularly useful since they serve as a basis for recovery of predicate argument structure and an answer to the question of who did what to whom. The Stanford labelled dependency scheme (de Marneffe and Manning 2008) has been used in many NLP applications including question answering, information extraction and sentiment analysis. The Stanford dependencies were originally designed to be produced from the output of phrase structure parsers but they have been used recently in the context of direct parsing into dependency trees using parsers such as Malt (Nivre, Hall, and Nilsson 2006) in research described in Cer et al. (2010), Petrov et al. (2010) and Çetinoğlu et al. (2010).

We train Malt to produce basic Stanford dependencies. We first train a model on the Wall Street Journal (WSJ) section of the Penn Treebank (Marcus et al. 1994), and we examine the parser's performance on a small treebank of sentences taken from microblogs (tweets). We find that the parser's performance drops 20 percentage points in labelled attachment accuracy. Because Malt accepts as input a sequence of part-of-speech (POS) tags rather than a sequence of words we also evaluate the accuracy of SVMTool (Giménez and Màrquez 2004), the POS tagger that we use to supply the input to Malt. A substantial proportion of the parsing errors can be attributed to POS tagging errors.

Unsupervised approaches to parser domain adaptation have met with moderate success in the last five years. McClosky et al. (2006) showed that the performance of the Charniak and Johnson two-stage reranking parser (Charniak and Johnson 2005) could be improved on out-of-domain text by retraining the first stage generative parser on trees produced by the two-stage parser. Petrov et al. (2010) demonstrated that the performance of a deterministic dependency parser on question data could be greatly improved upon by retraining it on a combination of its original material and trees produced by a slower, yet more accurate phrase structure parser. We combine these two ideas by retraining Malt on trees produced by a self-trained version of the Charniak and Johnson parser, achieving an LAS improvement of 4.67% on our Twitter test set. We examine in detail the effect of this uptraining on individual dependency relations.

Twitter

Twitter is a service that combines microblogging and social networking: through it, users can send short messages of up to 140 characters called *tweets* to their *followers*, i. e. other users who previously connected with the sender (social networking). The system is open as the connections are established without confirmation of the followee. Furthermore, the messages are publicly shown on the senders's profile page (microblogging). The service started as a "mobile status update service" but soon was widely used to report on events.[1] Alternative routes to content are provided by a *search* function and lists of *trending topics* for the past minute, day and week.

The service provider, Twitter Inc., reports a quickly growing user base (460,000 new users daily as of March 2011) and an average of 140 million tweets per day. The Twitter interface makes it easy to forward a tweet to all followers with a *retweet (RT)*. The @-sign before a user name creates a link to the user's profile page. A tweet containing such a link is called a *mention*. Mentions are also used for *replies* which, by convention, start with "@user". Another special symbol in Twitter is the *hashtag* which marks keywords that categorise the tweet. To use more than a single word for a category label, users have to omit word spaces, e. g. *#BookWeek*.[2]

Wu et al. (2011) classify Twitter users as celebrities, media, organisations, bloggers and ordinary users and find limited interaction between the first four groups, except for bloggers who retweet 85 times more than ordinary users. A study by Pear Analytics classifies over 40% of tweets as "pointless babble".[3] Yet, breaking news often appears on Twitter before it reaches mainstream media.

.

Dataset

We create a small treebank of 519 syntactically annotated sentences taken from tweets. The source for these sentences is a corpus of 60 million tweets on 50 themes including politics, business, sport and entertainment, collected using the public Twitter API between February and May 2009 (Bermingham and Smeaton 2010). Some Twitter-specific characteristics of this corpus are provided in Table 1. The tweets in our treebank were split by hand into sentences, usernames were replaced by the generic string *Username* and urls were replaced by *Urlname*. We use 269 sentences as a development set, which we refer to as *TwitterDev*, and the remaining 250 as a testset, which we refer to as *TwitterTest*. Table 2 contains statistics on the treebank sentences.

The treebank sentences were first parsed automatically using an implementation of the Collins Model 2 generative statistical parser (Bikel 2004). They were then corrected by hand by one annotator, using as a reference the Penn Treebank bracketing guidelines (Bies et al. 1995) and the Penn Treebank trees themselves. Twitter-specific structures obviously do not appear in the Penn Treebank and a decision had to be made on how these should be annotated. Links, usernames and hash tags are all annotated as proper nouns inside a single word noun phrase. Links at the end of a tweet are attached to the verb in the same way an adverb occurring at the end of a sentence would be. The symbol **RT** is annotated as a noun within a single word noun phrase. The annotator went through the dataset twice, and a second annotator then annotated 10% of the sentences. Agreement on labelled bracketing between the two annotators is 95.8%. The disagreements involve fragments, interjections and multi-word expressions (see Table 3).

Mean characters per tweet	80.9
Mean words per tweet	14.7
Proportion containing link	23.0%
Proportion containing hashtag	5.4%
Proportion mentions	38.9%
Proportion replies	31.6%
Proportion retweets	2.6%

Table 1: Twitter-specific characteristics of the full Twitter corpus

Corpus Name	***#Sen***	***SL Mean***	***SL Med.***	σ
TwitterDev	269	11.14	10	6.43
TwitterTest	250	11.36	10	6.80

Table 2: Basic Statistics on the Twitter treebank: number of sentences, average sentence length, median sentence length and standard deviation

Evaluation of WSJ-Trained Resources

In this section, we evaluate the accuracy of the POS tagger, SVMTool (Giménez and Màrquez 2004), and the dependency parser, Malt (Nivre, Hall, and Nilsson 2006), on the sentences in *TwitterDev*. As well as reporting tagging and parsing accuracy for *TwitterDev*, we also report performance on Section 22, *WSJ22*, as our in-domain reference test set. We also carry out a qualitative evaluation using those sentences from *TwitterDev* that are listed in Table 4.

Accuracy of POS Tagging

SVMTool (Giménez and Màrquez 2004) uses support vector machine learning to induce taggers for various languages. We use the WSJ-trained model supplied with the software. The accuracy of SVMTool on *TwitterDev* is 84.1% compared to an accuracy of 96.3% on *WSJ22*. The most common POS confusions for *TwitterDev* are listed in Table 5.

A substantial proportion of the errors are mistaggings of proper nouns. Some of these cases relate to the generic names *Username* and *Urlname*, which were used to replace usernames and links and which should both be tagged as

[1]Most information in this section is compiled from articles from `http://blog.twitter.com/` and `http://support.twitter.com/`.

[2]`http://www.newyorker.com/online/blogs/susanorlean/2010/06/hash.html`

[3]`http://www.pearanalytics.com/blog/wp-content/uploads/2010/05/Twitter-Study-August-2009.pdf`

(FRAG **(INTJ (UH congrats))** (NP (NNP Tiger)) (. !) (. !))
versus
(FRAG **(NP (NNS congrats))** (NP (NNP Tiger)) (. !) (. !))
(FRAG **(INTJ (IN Of) (NN course))** (. !))
versus
(FRAG **(PP (IN Of) (NP (NN course)))** (! !))
(S (VP (VBG picking) (PRT (RP up)) (NP (PRP$ my) (NN truck))
(PP (IN from) (NP (NNP toyota))) **(PRN (NP (JJ nice) (NNS folks))))**)
versus
(S (VP (VBG picking) (PRT (RP up)) (NP (PRP$ my) (NN truck))
(PP (IN from) (NP (NNP toyota)))) **(NP (JJ nice) (NNS folks))**)
(FRAG (NP (NNP USA)) (: -) (NP (NNP USA)) (: -) (NP (NNP USA)) (. !) (. !) (. !) (. !))
versus
(X (NP (NNP USA)) (: -) (NP (NNP USA)) (: -) (NP (NNP USA)) (. !) (. !) (. !) (. !))
(**FRAG** (NP (NNP Username)) **(INTJ (UH Okay) (, ,) (UH okay))** (. .))
versus
(**X** (NP (NNP Username)) **(ADJP (JJ Okay)) (, ,) (ADJP (JJ okay))** (. .))

Table 3: Inter-annotator disagreements on a subset of *TwitterDev* trees

1. I just think he looks like a big baby , and ppl USED to call him that.
I PRP just RB think VBP he PRP looks VBZ like IN a DT big JJ baby NN , ,
and CC ppl **NN** USED VBD to TO call VB him PRP that DT . .
2. been playing with the new Canon EOS 500d and the Nikon D5000 over the weekend .
been VBN playing VBG with IN the DT new JJ Canon NNP EOS NNP 500d **JJ**
and CC the DT Nikon NNP D5000 **NN** over IN the DT weekend NN . .
3. On Fox : RNC chair sends letter to GOP calling Obama " ARROGANT " #tcot #sgp #hhrs
On IN Fox NNP : : RNC NNP chair NN sends VBZ letter NN to TO GOP NNP
calling VBG Obama NNP `` `` ARROGANT **NNP** '' '' #tcot **NN** #sgp **NN** #hhrs **NNS**
4. FF > S4
FF **NN** > **NN** S4 **NN**
5.LOL !
LOL **NNP** ! .
6. i heart beltran .
i **FW** heart **NN** beltran **NN** . .
7. Man Utd through to the last 8 ...
Man NNP Utd NNP through IN to TO the DT last JJ 8 CD ... :
8. Bed soon .
Bed **VBN** soon RB . .
9. twas okay .
twas **NNS** okay JJ . .
10. Obama Loses Two More Appointees : Sanjay Gupta , Annette Nazareth Urlname
Obama NNP Loses VBZ Two CD More JJR Appointees **NNPS**
: : Sanjay NNP Gupta NNP , , Annette NNP Nazareth NNP Urlname NNP

Table 6: Output of SVMTool for examples from Table 4

1. I just think he looks like a big baby
, and ppl USED to call him that .
2. been playing with the new Canon EOS 500d
and the Nikon D5000 over the weekend .
3. On Fox : RNC chair sends letter to GOP
calling Obama " ARROGANT " #tcot #sgp #hhrs
4. FF > S4
5. LOL !
6. i heart beltran .
7. Man Utd through to the last 8 ...
8. Bed soon .
9. twas okay .
10. Obama Loses Two More Appointees
: Sanjay Gupta , Annette Nazareth Urlname

Table 4: Examples from *TwitterDev*

Gold/System	Freq.	Gold/System	Freq.
NNP/NN	59	VBZ/NNS	8
NN/NNP	54	UH/NNP	7
NNP/JJ	29	RB/NN	7
NNP/VB	10	NNP/CD	7
JJ/NN	10	NN/VB	6
UH/NN	8	VB/NN	6
JJ/NNP	8	VB/NNP	6
NNP/NNS	8	VBP/VB	6
NNPS/NNS	8	RP/IN	6

Table 5: SVMTool Mistaggings on *TwitterDev*

`NNP`. 19 of the 43 occurrences of *Username* are tagged incorrectly compared to just 3 of the 37 occurrences of *Urlname*. Another reason for the low recall of `NNP` tags is that tweeters, unlike Wall Street Journal editors, often do not capitalise proper nouns (see example 6 in Table 4). Hash tags should also be tagged as proper nouns — 7 of the 14 hash tags in *TwitterDev* have been mistagged. Apart from proper nouns beginning with lowercase characters, there are other capitalisation conventions that are worth mentioning because they are likely to be contributing towards the POS mistagging rate. One of these is the use of uppercase characters in an entire word or even sentence (see Examples 1 and 3 in Table 4). Foster (2010) identifies the use of uppercase characters as one of the factors in the relatively poor performance of parsers on sentences from a discussion forum. Inspecting the words mistagged by SVMTool, it seems that this is also a problem for SVMTool. We can see from Table 5 that some of the errors involve words that are not proper nouns being tagged as such. A possible reason for this, is that, in some tweets, news headlines in particular, the first character in every word is capitalised (see example 10 in Table 4).

A tagger's job is made more difficult if the word to be tagged is not in its lexicon. In this situation, the tagger can use clues based on the word's morphology, its position within the sentence and properties of words occurring very infrequently in its lexicon. Unsurprisingly, the unknown token rate in *TwitterDev* is much higher than in *WSJ22*: 16.6% compared to 2.8%. Excluding instances of *Username* and

Parser	LAS	UAS	LAS	UAS
	WSJ22		*TwitterDev*	
Malt Predicted Tag	87.98	90.61	67.64	73.75
Malt Gold Tag	89.95	91.61	78.67	81.86

Table 7: Malt Labelled Attachment and Unlabelled Attachment with SVMTool-tagged input and gold-tag input

Urlname, the proportion of unknown tokens in *TwitterDev* is 14.0%. Of the words mistagged by SVMTool, 53.2% are words that are unknown to the tagger.

We end this section by presenting, in Table 6, the output of SVMTool for our example sentences in Table 4. Tagging errors are highlighted in bold.

Accuracy of WSJ-trained Malt

Malt (Nivre, Hall, and Nilsson 2006) is a widely used multilingual parsing system. During training, a classifier is induced to predict a parsing action at a particular parsing configuration using information from the parse history and the remaining input string. During parsing, the classifier is used to drive the deterministic construction of a dependency tree. Malt can be used with several parsing algorithms including variants of shift-reduce parsing. We use the *stackeager* algorithm described in Nivre et al. (2009) and we train a linear classifier where the feature interactions are modelled explicitly. We train Malt on a version of Sections 2-21 of the WSJ treebank that has been converted to labelled dependency trees using the Stanford constituency to dependency converter. We use SVMTool to supply the POS tagged input to Malt.

Table 7 shows the labelled attachment accuracy (LAS) and unlabelled attachment accuracy (UAS) of a WSJ-trained Malt model on both *TwitterDev* and *WSJ22*. There is a very large difference in accuracy between the in-domain and out-of-domain test sets — an absolute difference of 20.34% in LAS. POS tagging errors account for a substantial proportion of this as the difference between automatic and gold tag input on *TwitterDev* is 11.03%. The dependency trees for two of the example sentences (sentences 1 and 6 from Table 4) are shown in Figures 1 and 2. The POS tagging error that occurs in Sentence 1 (*ppl* tagged as `NN` rather than `NNS`) does not prevent Malt from attaching *ppl* to the correct head using the correct dependency type. However, there is another misparse which is not caused by a POS tagging error, namely, the attachment of the last word *that* to the word *him* rather than *call*. The dependency tree in Figure 2 is completely misparsed due to the errors in POS tagging. *beltran* is incorrectly analysed as the head of the sentence, with *i* and *heart* incorrectly identified as nominal modifiers.

Improving Parser Performance

Malt Uptraining

Petrov et al. (2010) demonstrate that the Malt's performance on question data can be substantially improved by training it on trees produced by the Berkeley parser, which has the advantage of producing slightly more accurate Stanford depen-

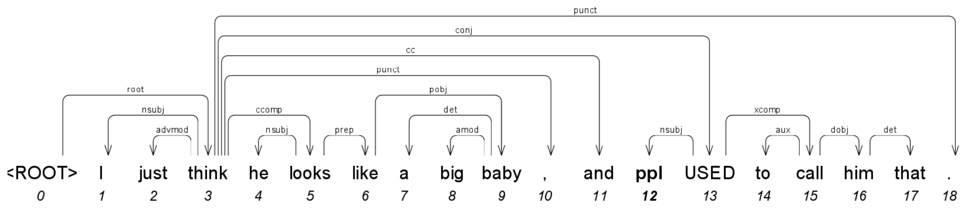

Figure 1: The baseline Malt analysis for Sentence 1 from Table 4

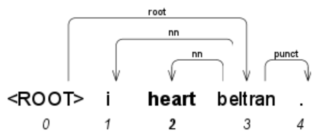

Figure 2: The baseline Malt analysis for Sentence 6 from Table 4

dency trees than Malt, but the disadvantage of being slower. Apart from exploring a different dataset, our twist on Petrov et al.'s (2010) work is to use as our phrase structure parser the even more accurate Charniak and Johnson two stage parser (Charniak and Johnson 2005) (we call this *vanilla uptraining*), and, in a second experiment, to use a self-trained version of this parser (we call this *domain-adapted uptraining*). The additional training sentences come not from Twitter but from a sports discussion forum.[4] An important point to note is that the POS tagger SVMTool also needs to be retrained on the same sentences as Malt.

The uptraining LAS results for *TwitterDev* are shown in Figure 3. The x-axis shows the amount of additional Charniak and Johnson parse trees that were added to either one or two copies of *WSJ2-21*. We can see from these results that both types of uptraining improve over the baseline but that domain-adapted uptraining is more successful than vanilla uptraining. Using the best model for *TwitterDev*, we parse the sentences in *TwitterTest* and achieve an LAS improvement of 4.67% and a UAS improvement of 3.74%. Both improvements are statistically significant.

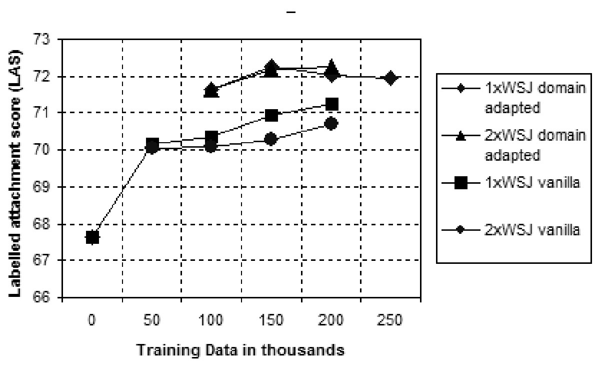

Figure 3: Malt uptraining results on *Twitterdev*

Uptraining Error Analysis

The performance on dependency types is detailed below:

amod The scores for adjectival modifiers are around 70%, following the baseline $<$ vanilla uptraining $<$ domain-adapted uptraining trend (69.4, 70.3, 72). Even the best score is lower than the gold-POS-tagged baseline (87%).

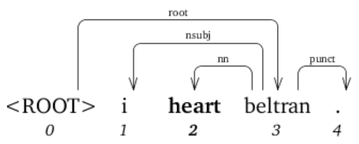

Figure 5: The domain adapted Malt analysis for Sentence 6 from Table 4

[4]A series of retraining experiments with a variety of parsers demonstrated that the discussion forum data (`http://news.bbc.co.uk/sport2/hi/606/default.stm`) was more useful than Twitter data as a source of additional unlabelled training data.

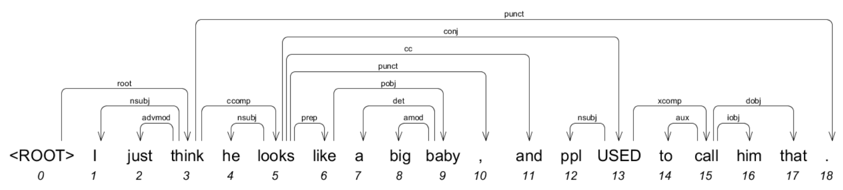

Figure 4: The domain adapted uptrained Malt analysis for Sentence 1 from Table 4

cc and conj Coordination is represented using two dependency relations. The first conjunct is the head of the coordination and the other conjuncts are dependent on the head via a conj relation. The coordinating item (e.g., *and*, *or*) is dependent on the head via the cc relation. Figure 1 contains such a coordinated phrase. Our experiments show that it is hard for parsers to recover the conj relation: the f-score starts at 54.8%, goes up to 67.5% in vanilla uptraining, then decreases to 65% in domain-adapted uptraining. cc is easier to recover. The baseline is 70.3%. vanilla uptraining results in a big jump to 81.6%. Malt benefits from the domain-adapted parse trees a lot and the f-score goes up to 85.3%. Both vanilla uptraining and domain-adapted uptraining are better than the gold-POS-tagged baseline, which is 79%.

ccomp Using domain-adapted trees in training also helps recover clausal complements. The baseline is 56.5%. There is a 1% absolute increase in vanilla uptraining, and a 7% absolute jump in domain-adapted uptraining.

cop Similar to ccomp, there is a 1 point increase in vanilla uptraining and a 6 point increase in domain-adapted uptraining. Still the best score is almost 6 points lower than the gold-POS-tagged baseline.

dep This relation is used rather when the converter cannot determine the dependency type and consequently it is very hard for parsers to correctly identify it. For the baseline system, the f-score is 15.8%, for vanilla uptraining it goes down to 13.9%, and for domain-adapted uptraining it is 17%.

dobj For direct objects, vanilla uptraining (75.2%) outperforms the baseline (65.2%) and domain-adapted uptraining (72%). The best system is 5% absolute lower than the gold-POS-tagged baseline.

neg Negation benefits highly from training data with domain-adapted trees. Training with vanilla trees does not affect the baseline f-score, both are 78.6%. When domain-adapted trees are used, the f-score jumps to 89.3%.

nn The relation nn represents the dependency of nouns to a head noun in an NP. The baseline 64.8% goes down slightly in vanilla uptraining and goes up slightly in domain-adapted uptraining. They are quite low when compared to the gold POS-tagged baseline which is 83.9%. Nominal compound Penn Treebank NPs are flat and this makes it hard for Stanford dependencies to correctly represent the dependency relation. Hence, the parsers trained on data converted by Stanford dependencies propagate the error.

nsubj Nominal subjects have an f-score of 69.4% in the baseline system. Vanilla uptraining adds a 4% absolute increment and domain-adapted uptraining uptraining adds a further 2.2% with a final f-score of 75.7%.

prep Uptraining helps find prepositional modifers by 3% absolute but there is only a slight increase from vanilla uptraining to domain-adapted uptraining.

xcomp Open clausal complements have a baseline score of 65.9%, which goes up to 81.1% with vanilla uptraining, but drops to 78.4% with domain-adapted uptraining. Both grammars outperform the gold POS-tagged baseline.

For sentences 1 and 6 from Table 4, the dependency trees produced by domain-adapted training of Malt are given in Figures 4 and 5. *ppl* in Sentence 1 is now correctly identified as NNS by the retrained tagger. The correct dependency relation of *ppl* in the baseline system remains the same. In the uptrained version, *that* is dependent on the correct head *call*, however with the wrong deprel dobj. The dobj correctly attached in the baseline is now mislabelled as iobj in the uptrained Malt. Another difference between the baseline grammar and the grammar uptrained on domain adapted trees is in the conjuncts of the coordination. This is a genuinely ambiguous sentence. Although the baseline grammar interpretation in Figure 1 seems more probable, the parse in Figure 4 is also reasonable. In Sentence 6, the correct POS tagging of *i* as PRP leads to a better parse tree in domain-adapted uptraining. The nominal modifier (nn) in Figure 2 is correctly replaced with a nominal subject (nsubj) in Figure 5 although the other parsing errors remain.

Conclusions

We have examined the consequences of applying an off-the-shelf WSJ-trained POS-tagging and dependency parsing model to the language of Twitter. Encouragingly, unsupervised techniques go some of the way towards improving performance over the off-the-shelf baseline . However, much work remains to be done, given the noisy, diverse and constantly changing nature of Twitter. Our next step is to experiment with the Twitter-specific POS tagset and tagger described by Gimpel et al. (2011).

Acknowledgements

This research has been supported by the Enterprise Ireland Commercialisation Fund (CFTD/2007/229) and the Science Foundation Ireland (Grant 07/CE/ I1142) as part of the Centre for Next Generation Localisation (www.cngl.ie) at Dublin City University, School of Computing, and by the French Agence Nationale pour la Recherche, through the SEQUOIA project (ANR-08-EMER-013). We thank the reviewers for their helpful comments.

References

Bermingham, A., and Smeaton, A. 2010. Classifying sentiment in microblogs: Is brevity an advantage? In *Proceedings of CKIM*.

Bies, A.; Ferguson, M.; Katz, K.; and MacIntyre, R. 1995. Bracketing guidelines for Treebank II style, Penn Treebank Project. Technical Report Tech Report MS-CIS-95-06, University of Pennsylvania.

Bikel, D. 2004. Intricacies of collins parsing model. *Computational Linguistics* 30(4):479–511.

Çetinoğlu, O.; Foster, J.; Nivre, J.; Hogan, D.; Cahill, A.; and van Genabith., J. 2010. Lfg without c-structures. In *Proceedings of the Ninth International Workshop on Treebanks and Linguistic Theories*.

Cer, D.; de Marneffe, M.-C.; Jurafsky, D.; and Manning, C. D. 2010. Parsing to stanford dependencies: Trade-offs between speed and accuracy. In *Proceedings of LREC*.

Charniak, E., and Johnson, M. 2005. Course-to-fine n-best-parsing and maxent discriminative reranking. In *Proceedings of the 43rd ACL*.

de Marneffe, M.-C., and Manning, C. D. 2008. The stanford typed dependencies representation. In *Proceedings of the COLING Workshop on Cross-Framework and Cross-Domain Parser Evaluation*.

Foster, J. 2010. "cba to check the spelling" Investigating parser performance on discussion forum posts. In *Proceedings of HLT:NAACL*.

Giménez, J., and Màrquez, L. 2004. Svmtool: A general pos tagger generator based on support vector machines. In *Proceedings of LREC*, 43–46.

Gimpel, K.; Schneider, N.; OConnor, B.; Das, D.; Mills, D.; Eisenstein, J.; Heilman, M.; Yogatama, D.; Flanigan, J.; and Smith, N. A. 2011. Part-of-speech Tagging for Twitter: Annotation, Features and Experiments. In *Proceedings of ACL:HLT*.

Marcus, M.; Kim, G.; Marcinkiewicz, M. A.; MacIntyre, R.; Bies, A.; Ferguson, M.; Katz, K.; and Schasberger, B. 1994. The penn treebank: Annotating predicate argument structure. In *Proceedings of the 1994 ARPA Speech and Natural Language Workshop*, 114–119.

McClosky, D.; Charniak, E.; and Johnson, M. 2006. Reranking and self-training for parser adaptation. In *Proceedings of the 21st COLING/44th ACL*.

Nivre, J.; Hall, J.; and Nilsson, J. 2006. Maltparser: A data-driven parser-generator for dependency parsing. In *Proceedings of LREC*, 2216–2219.

Nivre, J.; Kuhlmann, M.; and Hall, J. 2009. An improved oracle for dependency parsing with online reordering. In *Proceedings of IWPT'09*, 73–76.

Petrov, S.; Chang, P.-C.; Ringgaard, M.; and Alshawi, H. 2010. Uptraining for accurate deterministic question parsing. In *Proceedings of EMNLP 2010*.

Wu, S.; Hofman, J.; Mason, W.; and Watts, D. 2011. Who says what to whom on twitter. In *Proceedings of the International World Wide Web Conference Committee (IW3C2)*.