#hardtoparse: POS Tagging and Parsing the Twitterverse

Jennifer Foster[1], **Özlem Çetinoğlu**[1], **Joachim Wagner**[1], **Joseph Le Roux**[2]
Stephen Hogan[3], **Joakim Nivre**[4], **Deirdre Hogan**[1] and **Josef van Genabith**[1]
[1,3]NCLT/CNGL, Dublin City University, Ireland
[2]LIF - CNRS UMR 6166, Université Aix-Marseille, France
[4]Department of Linguistics and Philology, Uppsala University, Sweden
[1]{jfoster,ocetinoglu,jwagner,dhogan,josef}@computing.dcu.ie
[2]joseph.le-roux@lif.univ-mrs.fr, [3]stephen.hogan2@mail.dcu.ie
[4]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 Twitter 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 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 communities. How can a tool trained on one domain 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 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) and Petrov 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 (LAS). 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 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 analyses produced for questions 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% 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 by the followee. Furthermore, the messages are publicly shown on the senders' 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, 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 test set, which we refer to as *TwitterTest*.

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).

From the full Twitter corpus, we also constructed a subcorpus to be used as a source of unlabelled training data. As with the treebank tweets, links were replaced by the term *Urlname* and usernames by *Username*. Tweets with more than one non-ASCII character were removed, and the remaining tweets were passed through our automatic sentence splitter and tokeniser, resulting in a corpus of 1,401,533 sentences. We refer to this as the *TwitterTrain* corpus. Table 2 contains statistics on *TwitterDev*, *TwitterTest* and *TwitterTrain*.

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	11.1	10	6.4
TwitterTest	250	11.3	10	6.8
TwitterTrain	1,401,533	8.6	7	6.1

Table 2: Basic Statistics on the Twitter training, development and test sets: 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 WSJ 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

[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 .
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*

usernames and links and which should both be tagged as 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 online shouting 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).

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*

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 sentential context 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 *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 learns to predict an action at a particular configuration using information from the parse history and the input string. During parsing, the classifier is used to deterministically construct 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

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

Parser	**LAS**	**UAS**	**LAS**	**UAS**
	WSJ22		*TwitterDev*	
Malt Predicted Tag	87.98	90.61	67.33	73.56
Malt Gold Tag	89.95	91.61	78.32	81.63

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

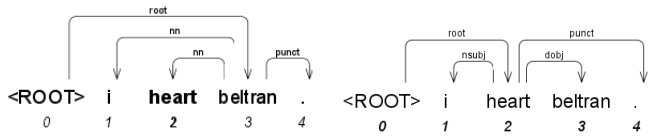

Figure 2: The baseline analysis (left) and domain-adapted uptrained analysis (right) for Ex. 6 from Table 4

Stanford dependency trees. 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 20.65% in LAS. POS tagging errors account for a substantial proportion of this as the difference between automatic and gold tag input on *TwitterDev* is 10.99%.

The dependency trees for two of the example sentences (sentences 2 and 6 from Table 4) are shown in Figures 1 and 2. Coordination is represented using two dependency relations, namely `cc` and `conj`. 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. The top tree in Figure 1 contains such a coordinated phrase, *the new Canon EOS 500d and the Nikon D5000*, which has been misanalysed by the parser, with *new* incorrectly identified as the first conjunct. Note that the correct head of the first conjunction, *500d*, has been mistagged as an adjective rather than a noun. The dependency tree on the left 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 questions can be substantially improved by training it on trees produced for question data by the Berkeley parser, which has the advantage of producing slightly more accurate Stanford dependency 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 (we call this *vanilla uptraining*), and, in a second experiment, to use a version of this parser that has itself been self-trained using the protocol described by McClosky et al. (2006) (we call this *domain-adapted uptraining*). We use as training data the sentences in the *TwitterTrain* corpus. Two disjoint subsets of this corpus are used: the sentences in one subcorpus are parsed with either the WSJ-trained or self-trained version of the Charniak and Johnson parser and added as training material to Malt,

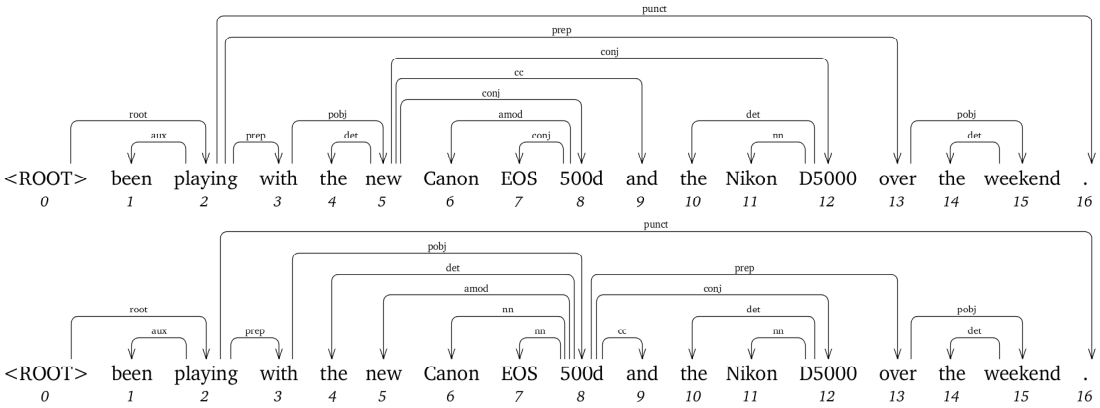

Figure 1: The baseline analysis (top) and domain-adapted uptrained analysis (bottom) for Ex. 2 from Table 4

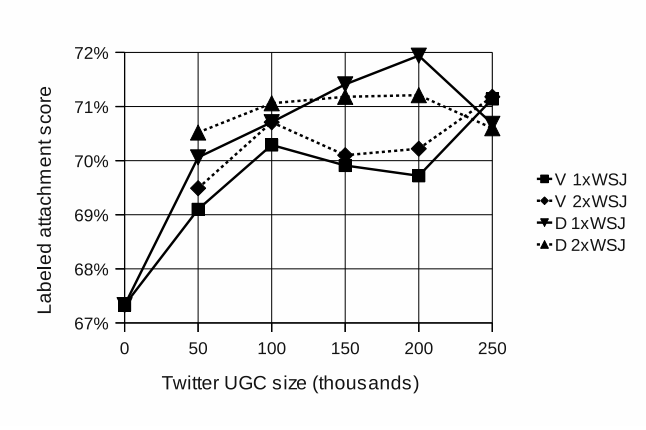

Figure 3: Malt uptraining results on *Twitterdev*: V stands for vanilla uptraining and D for domain-adapted uptraining

and the sentences in the other subcorpus are used for self-training the Charniak and Johnson parser for the domain-adapted version of uptraining. 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 (67.33%) but that domain-adapted uptraining (71.94%) is slightly more successful than vanilla uptraining (71.18%). Using the best model for *TwitterDev*, we parse the sentences in *TwitterTest* and achieve an LAS improvement of 4% and a UAS improvement of 2.5%. Both improvements are statistically significant.

Uptraining Error Analysis

There are 45 different Stanford dependency relations. In this section, we analyse the most frequent or linguistically most interesting ones. We compare the baseline, best vanilla uptraining, and best domain-adapted uptraining grammars. The results are shown in Table 8.

DepRel	Gold POS	Baseline	Vanilla	Domain-adapted
advmod	83.48	70.20	**74.53**	74.42
amod	87.31	**69.41**	63.48	69.09
aux	91.41	87.50	90.70	**91.05**
cc	78.95	70.27	**81.58**	81.08
ccomp	60.76	55.81	**59.77**	55.81
conj	65.88	54.76	**59.74**	**59.74**
cop	83.05	71.54	**77.31**	73.50
dep	26.9	15.86	**17.42**	16.83
det	96.25	92.45	92.79	**93.41**
dobj	80.37	65.19	70.29	**75.24**
neg	85.71	78.57	80.00	**85.18**
nn	83.86	**64.74**	64.72	64.63
nsubj	78.98	68.13	**75.22**	74.73
pobj	91.30	85.84	**87.72**	87.67
prep	85.40	75.50	**80.26**	80.00
root	74.40	62.98	69.65	**70.39**
xcomp	71.61	64.20	70.89	**76.32**

Table 8: *TwitterDev* f-scores of some dependency relations with different grammars: baseline with gold POS, baseline with predicted POS, best vanilla uptraining, and best domain-adapted uptraining.

The relations `aux`iliaries, `det`erminers, `neg`ation modifiers, `prep`ositions and objects of prepositions (`pobj`) are relatively easy to recover. The relations `aux`, `det` and `neg` follow the baseline $<$ vanilla $<$ domain-adapted trend. For `prep` and `pobj`, domain-adapted uptraining does not offer any improvement over vanilla uptraining.

The grammatical relations of nominal subject (`nsubj`) and direct object (`dobj`) both benefit from uptraining, with the latter getting a significant boost from domain-adapted uptraining. Clausal complements (`ccomp`) benefit from vanilla uptraining but not at all from domain-adapted trees. Open clausal complements (`xcomp`), on the other hand, benefit from vanilla uptraining and even more so from domain-adapted uptraining.

Our experiments show that it is harder for parsers to recover the conj relation than the cc relation. For both relations, domain-adapted uptraining does not improve over vanilla uptraining — the use of additional Twitter material in training the Charniak and Johnson parser is not helping its analysis of coordination. Note that for the cc relation, both vanilla uptraining and domain-adapted uptraining are better than the gold-POS-tagged baseline.

The only examples where uptraining does not help in improving parser accuracy are nn and amod. The relation nn represents the dependency of nouns to a head noun in an NP. The baseline goes down slightly in vanilla and domain-adapted uptraining and both scores are quite low when compared to the gold POS-tagged baseline. For adjectival modifiers (amod), the baseline decreases for vanilla uptraining and increases again to baseline levels for domain-adapted uptraining. The common POS tagging confusion between adjectives and nouns (see Table 5) is the likely culprit here since these tags are also commonly confused in the Charniak and Johnson trees.

The dep relation is used when the converter cannot determine the dependency type and consequently it is very hard for parsers to correctly identify, even with uptraining.

For sentences 2 and 6 from Table 4, the dependency trees produced by the domain-adapted uptraining of Malt are given in Figures 1 (bottom tree) and 2 (right tree). Here we see an example where the adjective/noun confusion has been corrected using uptraining. The word *500d* in Sentence 2 is now tagged as NN by the retrained tagger. This is not the correct tag (it should be NNP) but it is better than the previous tag of JJ and 500d is now identified as the head of the first conjunct, rather than *new* (see top tree). Note that in the uptrained analysis, a word that was attached correctly in the baseline is now incorrectly attached, namely, the preposition *over*. In Sentence 6, the correct POS tagging of *i* as PRP leads to a perfect parse tree in domain-adapted uptraining, even though the other two POS tagging errors remain: the nominal modifiers (nn) in the left tree in Figure 2 are correctly replaced with a nominal subject (nsubj) and direct object (dobj) in the right tree.

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 steps are to compare the two types of uptraining using larger training set sizes and 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.

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)*.