To Normalize, or Not to Normalize: The Impact of Normalization on Part-of-Speech Tagging

Rob van der Goot **Barbara Plank** **Malvina Nissim**
Center for Language and Cognition, University of Groningen, The Netherlands
{r.van.der.goot,b.plank,m.nissim}@rug.nl

Abstract

Does normalization help Part-of-Speech (POS) tagging accuracy on noisy, non-canonical data? To the best of our knowledge, little is known on the actual impact of normalization in a real-world scenario, where gold error detection is not available. We investigate the effect of automatic normalization on POS tagging of tweets. We also compare normalization to strategies that leverage large amounts of unlabeled data kept in its raw form. Our results show that normalization helps, but does not add consistently beyond just word embedding layer initialization. The latter approach yields a tagging model that is competitive with a Twitter state-of-the-art tagger.

1 Introduction

Non-canonical data poses a series of challenges to Natural Language Processing, as reflected in large performance drops documented in a variety of tasks, e.g., on POS tagging (Gimpel et al., 2011; Hovy et al., 2014), parsing (McClosky, 2010; Foster et al., 2011) and named entity recognition (Ritter et al., 2011). In this paper we focus on POS tagging and on a particular source of non-canonical language, namely Twitter data.

One obvious way to tackle the problem of processing non-canonical data is to build taggers that are specifically tailored to such text. A prime example is the ARK POS tagger, designed especially to process English Twitter data (Gimpel et al., 2011; Owoputi et al., 2013), on which it achieves state-of-the-art results. One drawback of this approach is that non-canonical data is not all of the same kind, so that for non-canonical non-Twitter data or even collections of Twitter samples from different times, typically a new specifically dedicated tool needs to be created.

JJ	NN	NN	NNS
new	pix	comming	tomoroe
JJ	NNS	VBG	NN
new	pictures	coming	tomorrow

Figure 1: Example tweet from the test data, raw and normalized form, tagged with Stanford NLP.

The alternative route is to take a general purpose state-of-the-art POS tagger and adapt it to successfully tag non-canonical data. In the case of Twitter, one way to go about this is *lexical normalization*. It is the task of detecting "ill-formed" words (Han and Baldwin, 2011) and replacing them with their canonical counterpart. To illustrate why this might help, consider the following tweet: "new pix comming tomoroe". An off-the-shelf system such as the Stanford NLP suite[1] makes several mistakes on the raw input, e.g., the verb 'comming' as well as the plural noun 'pix' are tagged as singular noun. Instead, its normalized form is analyzed correctly, as shown in Figure 1.

While being a promising direction, we see at least two issues with the assessment of normalization as a successful step in POS tagging non-canonical text. Firstly, normalization experiments are usually carried out assuming that the tokens to be normalized are already detected (*gold error detection*). Thus little is known on how normalization impacts tagging accuracy in a real-world scenario (*not* assuming gold error detection). Secondly, normalization is one way to go about processing non-canonical data, but not the only one (Eisenstein, 2013; Plank, 2016). Indeed, alternative approaches leverage the abundance of *unlabeled data* kept in its raw form. For instance,

[1] http://nlp.stanford.edu:8080/parser/index.jsp, accessed June 1, 2017.

such data can be exploited with semi-supervised learning methods (Abney, 2007). The advantage of this approach is that portability could be successful also towards domains where normalization is not necessary or crucial. These observations lead us to the following research questions:

Q1 In a real-world setting, without assuming gold error detection, does normalization help in POS tagging of tweets?

Q2 In the context of POS tagging, is it more beneficial to normalize input data or is it better to work with raw data and exploit large amounts of it in a semi-supervised setting?

Q3 To what extent are normalization and semi-supervised approaches complementary?

To answer these questions, we run a battery of experiments that evaluate different approaches. Specifically:

1. We study the impact of normalization on POS tagging in a realistic setup, i.e., we compare normalizing only unknown words, or words for which we know they need correction; we compare this with a fully automatic normalization model (Section 3).

2. We evaluate the impact of leveraging large amounts of unlabeled data using two approaches: a) deriving various word representations, and studying their effect for model initialization (Section 4.1); b) applying a bootstrapping approach based on self-training to automatically derive labeled training data, evaluating a range of a-priori data selection mechanisms (Section 4.2).

3. We experiment with combining the most promising methods from both directions, to gain insights on their potential complementarity (Section 5).

2 Experimental Setup

We run two main sets of POS tagging experiments. In the first one, we use normalization in a variety of settings (see Section 3). In the second one, we leverage large amounts of unlabeled data that does not undergo any normalization but is used as training in a semi-supervised setting (Section 4). For all experiments we use existing datasets as well as newly created resources, cf. Section 2.1. The POS model used is described in Section 2.2.

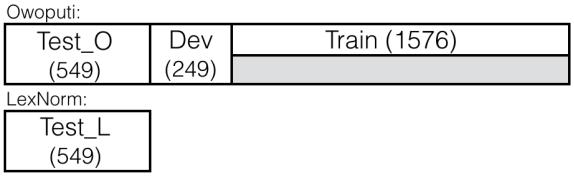

Figure 2: Labeled data for POS and normalization. Gray area: no gold normalization layer available.

2.1 Data

The annotated datasets used in this study originate from two sources: Owoputi et al. (2013) and Han and Baldwin (2011), which we will refer to as OWOPUTI and LEXNORM, respectively. All datasets used in this study are annotated with the 26 Twitter tags as described in (Gimpel et al., 2011).[2] OWOPUTI was originally annotated with POS labels, whereas LEXNORM was solely annotated for normalization. Li and Liu (2015) added a POS tag layer to the LEXNORM corpus, and a normalization layer to 798 Tweets from OWOPUTI, which we split into a separate DEV and TEST part of 249 and 549 Tweets, respectively, keeping the original POS labels. We use DEV throughout all experiments during development, and test our final best system on the held-out test sets (both containing 549 tweets). An illustration of the data is given in Figure 2.

For the different improvements to our baseline tagger, we need raw data from the target domain (Twitter). In addition, the normalization model needs unlabeled canonical data. We use a snapshot of English Wikipedia as unlabeled canonical data source. To get raw data for the social media domain, we collected Tweets during the whole year of 2016 by means of the Twitter API. We only collected Tweets containing one of the 100 frequent words in the Oxford English Corpus[3] as a rough language filter. This resulted in a dataset of 760,744,676 English Tweets. We do some very basic pre-processing in which we replace urls and usernames by <URL> and <USERNAME>, and remove duplicate tweets. Because of different casing strategies, we always apply a simple postprocessing step to 'rt' (retweet) tokens.

[2] Some tags are rare, like M and Y. In fact, M occurs only once in TEST_L; Y never occurs in DEV and only once in TEST_L and three times in TEST_O. Therefore our confusion matrices (over DEV and TEST_O, respectively) have different number of labels on the axes.

[3] https://en.wikipedia.org/wiki/Most_common_words_in_English

2.2 Model

We use BILTY, an off-the-shelf bi-directional Long Short-Term Memory (bi-LSTM) tagger which utilizes both word and character embeddings (Plank et al., 2016). The tagger is trained on 1,576 training tweets (Section 2.1). We tune the parameters of the POS tagger on the development set to derive the following hyperparameter setup, which we use throughout the rest of the experiments: 10 epochs, 1 bi-LSTM layer, 100 input dimensions for words, 256 for characters, σ=0.2, `constant` embeddings initializer, `Adam` trainer, and updating embeddings during backpropagation.[4]

3 To Normalize

First we evaluate the impact of normalization on the POS tagger.

3.1 Model

We use an in-house developed normalization model (van der Goot and van Noord, 2017).[5] The model is based on the assumption that different normalization problems require different handling. First, since unintentional disfluencies can often be corrected by the use of a spell checker, the normalization model exploits Aspell[6]. Second, since intentional disfluencies typically have a much larger edit distance, the normalization system uses word embeddings (Mikolov et al., 2013);[7] words close to the non-canonical word in the vector space are considered potential normalization candidates. On top of that, the model uses a lookup list generated from the training data, which works especially well for slang.

Features originating from the ranking are combined with uni- and bi-gram probabilities from Wikipedia data as well as from raw Tweets (Section 2.1). A random forest classifier (Breiman, 2001) is then used to rank the candidates for each word. Note that the original word is also a candidate; this enables the model to handle error detection, which is not always the case in models of previous work.

We train the normalization model on 2,577 tweets from Li and Liu (2014). Our model (van der Goot and van Noord, 2017) achieves state-of-art performance on the erroneous tokens (using gold error detection) on the LexNorm dataset (Han and Baldwin, 2011) as well as state-of-art on another corpus which is usually benchmarked without assuming gold error detection (Baldwin et al., 2015). We refer the reader to the paper (van der Goot and van Noord, 2017) for further details.

To obtain a more detailed view of the effect of normalization on POS tagging, we investigate four experimental setups:

- normalizing only unknown words;
- considering all words: the model decides whether a word should be normalized or not;
- assuming gold error detection: the model knows which words should be normalized;
- gold normalization; we consider this a theoretical upper bound.

Traditionally, normalization is used to make the test data more similar to the train data. Since we train our tagger on the social media domain as well, the normalization of only the test data might actually result in more distance between the train and test data. Therefore, we also train the tagger on normalized training data, and on the union of the normalized and the original training data.

3.2 Results

The effects of the different normalization strategies on the DEV data are shown in Table 1. Throughout the paper we report average accuracies over 5 runs including standard deviation.

The first row shows the effect of normalization at test-time only. From these results we can conclude that normalizing all words is beneficial over normalizing only unknown words; this shows that normalization has a positive effect that goes beyond changing unknown words.

The results of using the gold normalization suggest that there is still more to gain by improving the normalization model. In contrast, the results for gold error detection (GOLDED) show that error detection is not the main reason for this difference, since the performance difference between ALL and GOLDED is relatively small compared to the gap with GOLD.

[4] `Adam` was consistently better than `sgd` on this small training dataset. More LSTM layers lowered performance.

[5] Available at: https://bitbucket.org/robvanderg/monoise

[6] www.aspell.net

[7] Using the tweets from Section 2.1 and the following parameters: `-size 400 -window 1 -negative 5 -sample 1e-4 -iter 5`

↓ Train → Test	RAW	UNK	ALL	GOLDED	GOLD
RAW	82.16 (±.33)	83.44 (±.25)	**84.06** (±.32)	84.67 (±.23)	86.71 (±.25)
ALL	80.42 (±.71)	81.99 (±.64)	83.87 (±.28)	84.05 (±.31)	86.11 (±.14)
UNION	81.54 (±.27)	83.11 (±.31)	84.04 (±.34)	84.42 (±.24)	86.35 (±.17)

Table 1: Results of normalization (N) on DEV (macro average and stdev over 5 runs). RAW: no normalization, ALL: automatic normalization, UNK: normalize only unknown words, GOLDED: use gold error detection, GOLD: use gold normalization (Oracle). Row: whether training data is normalized. UNION stands for the training set formed by the union of both normalized and original raw data.

Considering the normalization of the training data, we see that it has a negative effect. The table suggests that training on the raw (non-normalized) training data works best. Adding normalized data to raw data (UNION) does not yield any clear improvement over RAW only, but requires more training time. For the test data, normalization is instead always beneficial.

To sum up, normalization improved the base tagger by 1.9% absolute performance on the development data, reaching 84.06% accuracy. Overall, our state-of-art normalization model only reaches approximately 50% of the theoretical upper bound of using gold normalization. We next investigate whether using large amounts of unlabeled data can help us to obtain a similar effect.

4 Or Not to Normalize

An alternative option to normalization is to leave the text as is, and exploit very large amounts of raw data via semi-supervised learning. The rationale behind this is the following: provided the size of the data is sufficient, a model can be trained to naturally learn the POS tags of noisy data.

4.1 Effect of Word Embeddings

An easy and effective use of word embeddings in neural network approaches is to use them to initialize the word lookup parameters.

We train a skip-gram word embeddings model using word2vec (Mikolov et al., 2013) on 760M tweets (as described in Section 3.1). We also experiment with structured skip-grams (Ling et al., 2015), an adaptation of word2vec which takes word order into account. It has been shown to be beneficial for syntactically oriented tasks, like POS tagging. Therefore we want to evaluate structured skip-grams as well.

The normalization model uses word embeddings with a window size of 1; we compare this with the default window size of 5 for structured skip-grams.

Results Table 2 shows the results of using the different skip-gram models for initialization of the word embeddings layer. Structured skip-grams perform slightly better, confirming earlier findings. Using a smaller window is more beneficial, probably because of the fragmented nature of Twitter data.

Structured skip-grams of window size 1 result in the best embedding model. This results in an improvement from 82.16% (Table 1) to 88.51% accuracy. This improvement is considerably larger than what obtained by normalization (82.16).

4.2 Effect of Self-training

We work with a rather small training set, which is all that is available to us in terms of gold data. This is due to the use of an idiosyncratic tagset (Gimpel et al., 2011). Adding more data could be beneficial to the system. To get an idea of how much effect extra annotated data could potentially have on POS tag accuracy, we plot the performance using smaller amounts of gold training data in Figure 3. We can see that there is still a slight upward trend; however, even when adding manually annotated data, the performance sometimes drop, especially after adding 55% of the training data.

To create more training data, we use an iterative indelible self-training setup (Abney, 2007) to obtain automatically labeled data. Specifically: 100

	WINDOW SIZE	
	1	5
SKIPG.	88.14 (±.30)	87.56 (±.08)
STRUCT.SKIPG.	**88.51** (±.24)	88.11 (±.49)

Table 2: Accuracy on raw DEV: various pre-trained skip-gram embeddings for initialization.

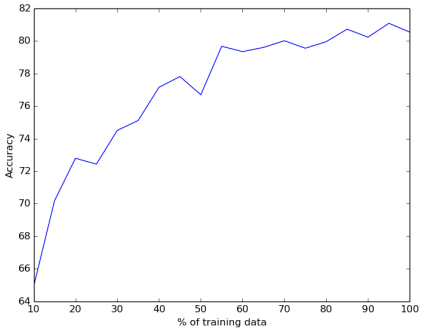

Figure 3: Effect of increasing amounts of training data (100% training data == 1,576 tweets).

tweets are tagged, they get added to the training data, and after this a new model is trained.

While we do not adopt any filtering strategy on the predictions (e.g., confidence thresholds), we do explore different strategies of *a-priori* data selection, from two corpora: raw tweets (Section 3.1), and the English Web Treebank (Petrov and McDonald, 2012).

For the English Web Treebank (EWT), we directly use raw text. Moreover, because the texts in the EWT are distributed by domains, i.e., *answers*, *emails*, *weblogs*, *newsgroups*, and *reviews*, we preserve this information and keep the data separate according to their domain to see whether adding data from the different domains can provide a more useful signal.

For the raw tweets, we compare different strategies of sampling. In addition to selecting random tweets, we experimented with selections aimed at providing the tagger with specific information that we knew was missing or confusing in the original training data. One strategy thus was to include tweets that contained words occurring in the development data but not in the training data. Note that this would result in a very slow tagger in a real-world situation, since the tagger needs to be retrained for every new unknown word. Another strategy was based on a preliminary analysis of errors on the development data: from the confusion matrix we observed that a frequently confounded tag was proper noun. Considering named entities as adequate proxies for proper nouns in this context, we also experimented with adding tweets that contained named entities. The detection of named entities was performed using a Twitter-specific named entity recognizer (Ritter et al., 2011). For control and comparison, we also collect additional training data where only tweets that do *not* contain named entities are selected. Hence, we end up with the following four sampling strategies:

- random sampling
- tweets containing words which occur in the development data, but not in the training data
- tweets containing named entities
- tweets not containing named entities

Results Adding more automatically-labeled data did not show any consistent improvement. This holds for both selection methods regarding named entities (presence/absence of NERs) and different domains of the Web treebank. Therefore we do not elaborate further here. We hypothesize that post-selection based on e.g., confidence sampling, is a more promising direction. We consider this future work.

5 Normalizing and Not Normalizing

In the previous sections, we explored ways to improve the POS tagging of Tweets. The most promising directions were initializing the tagger with pre-trained embeddings and using normalization. Self-training was not effective. In this Section, we report on additional experiments on the development data aimed at obtaining insights on the potential of combining these two strategies.

5.1 Consequences of Normalization

	BILTY	+NORM	+VECS	+COMB
CANONICAL	86.1	85.6	91.2	90.1
NON-CANON.	50.8	70.3	71.1	78.5

Table 3: Effect of different models on canonical/non-canonical words.

Table 3 shows the effect of the two approaches on the two subsets of tokens (canonical/non-canonical) on the DEV set. Word embeddings have a higher impact on standard, canonical tokens. It is interesting to note that word embeddings and normalization both have a similar yet complementary effect on the words to be normalized (non-canonical). The improvements on non-canonical words seem to be complementary. The combined model additionally improves on words which need normalization, whereas it scores almost 1% lower on canonical words. This suggests that both strategies have potential to complement each other.

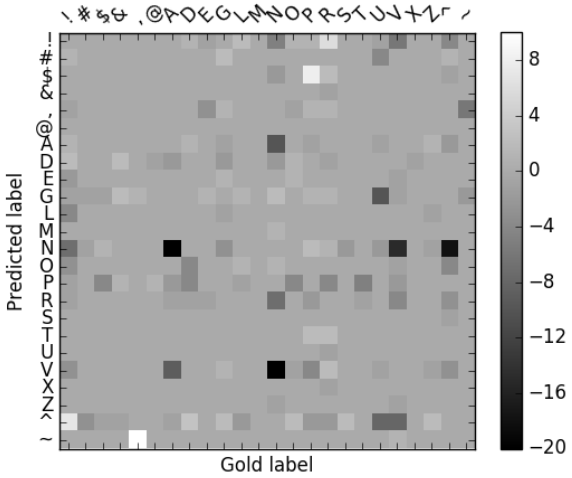

Figure 4: Differences in numbers of errors on development data between best normalization setting and best word embeddings. Dark means normalization makes more errors.

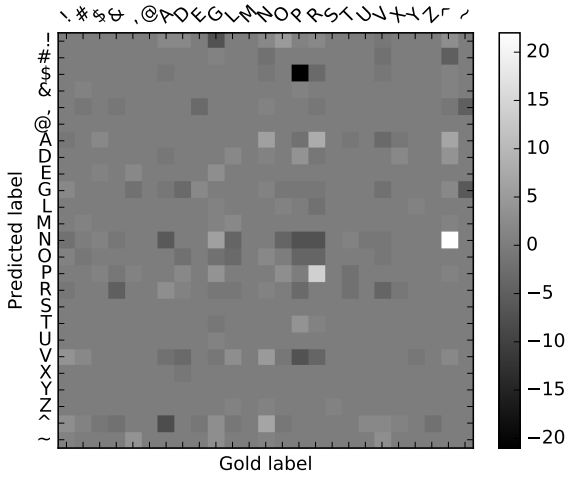

Figure 5: Comparison of errors per POS between our best model and the ARK tagger on TEST_O; darker means our system performs better.

5.2 Performance per POS

We compare the type of errors made by the best normalization setting versus the best word embeddings setting in a confusion matrix which displays the difference in errors in Figure 4. To recall: the best normalization setting was to use the raw training data, normalizing all words at test time; the best word embeddings model was a structured skip gram embeddings model with a window of 1.

In the confusion graph it becomes clear that normalization results in over-predicting nouns (N), which often get confused with verbs (V), adjectives (A) and proper nouns (^). Normalization is better at recognizing prepositions (P), which it confuses less with numerals ($) compared to the embedding model. This is due to normalizing '2' and '4'. Instead, the embedding model has better predictions for proper nouns, nouns and verbs, presumably due to the higher coverage.

6 Evaluation

In this section we report results on the test data, as introduced in Section 2.1.

Our main aim is to compare different approaches for successfully applying a generic state-of-the-art POS tagger to Twitter data. Therefore we have to assess the contribution of the two methods we explore (normalization and using embeddings) and see how they fare, not only to each other but also in comparison to a state-of-the-art Twitter tagger. We use the ARK tagger (Owoputi et al., 2013) and retrain it on our dataset for direct comparison with our models. The ARK system is a conditional random fields tagger, which exploits clusters, lexical features and gazetteers.

Table 4 shows the performance of our best models and the ARK tagger on the test datasets.

Embeddings work considerably better than normalization, which confirms what we found on the DEV data. The combined approach yields the highest accuracy over all evaluation sets, however, it significantly differs from embeddings only on TEST_L. This can be explained by our earlier observation (cf. Table 3), which shows that COMB yields the highest improvement on non-canonical tokens, but the same does not hold for canonical tokens. Notice that TEST_L does indeed contain the highest proportion of non-canonical tokens.

Our best results on all datasets are comparable to the state-of-the-art results achieved by the ARK tagger. In Figure 5 we compare the errors made by our system (COMB in Table 4) and ARK on TEST_O, which is the test set on which both tag-

	DEV	TEST_O	TEST_L
% non-canonical tokens	11.75%	10.95%	12.09%
BILTY	82.16 (±.33)	83.81 (±.23)	80.78 (±.32)
+NORM	84.06 (±.32)	84.73 (±.19)	84.61 (±.21)
+EMBEDS	**88.51** (±.24)	**90.02** (±.35)	88.53 (±.41)
+COMB	**88.89** (±.25)	**90.25** (±.19)	**89.63** (±.13)
ARK	89.08	90.65	89.67

Table 4: Results on test data (average over 5 runs) compared to ARK-tagger (Owoputi et al., 2013). Bold: best result (in case of multiple: no stat.significant difference according to randomization test).

gers obtain the highest performance.

The ARK tagger has difficulties with prepositions (P), which are mistagged as numerals ($). These are almost all cases of '2' and '4', which represent Twitter slang for 'to' and 'for', respectively. Our system performs a lot better on these, due to the normalization model as already observed earlier. Still regarding prepositions, ARK is better at distinguishing them from adverbs (R), which is a common mistake for our system. Our tagger makes more mistakes on confusing proper nouns (^) with nouns (N) in comparison to ARK.

7 Related Work

Theoretically, this works fits well within the debate on normalization vs domain adaptation (Eisenstein, 2013). For a practical comparison, the work most related to ours is that of Li and Liu (2015). They propose a joint model for normalization and POS tagging. The candidate lists of six different normalization models, including spell checkers and machine translation systems, are combined with all their possible POS tags as found by the ARK Twitter POS tagger. Note that they use gold error detection, while we perform fully automatic normalization. These combined units of words and POS tags are then used to build joint Viterbi decoding (Viterbi, 1973). The optimal path in this decoding does not only contain a sequence of normalized tokens, but also a sequence of POS tags. This joint model proves to be beneficial for both tasks.

Work on normalization for improving POS tagging has also been done on other languages. For example, Ljubešić et al. (2017) show that performing normalization, in addition to using external resources, can remove half of the errors of a standard POS tagger for South Slavic languages. Quite surprisingly, instead, of all systems participating in shared tasks on POS tagging of Twitter data for both Italian (Bosco et al., 2016) and German (Beißwenger et al., 2016), none of the participating systems incorporated any normalization strategy before performing POS tagging.

Finally, normalization for POS tagging is certainly not limited to non-canonical data stemming from social media. Indeed, another stream of related work is focused on historical data, usually originating from the 15th till the 18th century. The motivation behind this is that in order to apply current language processing tools, the texts need to be normalized first, as spelling has changed through time. Experiments on POS tagging historical data that was previously normalized have been investigated for English (Yang and Eisenstein, 2016), German (Bollmann, 2013), and Dutch (Hupkes and Bod, 2016; Tjong Kim Sang, 2016). In this latter work, different methods of 'translating' historical Dutch texts to modern Dutch are explored, and a vocabulary lookup-based approach appears to work best.[8] In this paper we focused on normalization and POS tagging for Twitter data only.

8 Conclusion

We investigated the impact of normalization on POS tagging for the Twitter domain, presenting the first results on automatic normalization and comparing normalization to alternative strategies. We compared a generic tagger to a tagger specifically designed for Twitter data.

Regarding Q1, we can conclude that normalization does help. However, using large amounts of unlabeled data for embedding initialization yields an improvement that is twice as large as the one

[8] Interestingly, this work also resulted in a shared task on normalization of historical Dutch, in which the secondary evaluation metric was POS tagging accuracy: https://ifarm.nl/clin2017st/.

obtained using normalization (Q2).

Combining both methods (Q3) does indeed yield the highest scores on all datasets. This suggests that the two approaches are complementary, also because in isolation their most frequent errors differ. However, the contribution of normalization on top of embeddings alone is relatively small and only significant on one test set, which was specifically developed for normalization and contains the largest proportion of non-canonical tokens.

Overall, our best model is comparable to the ARK tagger. As a general direction, our results suggest that exploiting large amounts of unlabeled data of the target domain is preferable. However, if the data is expected to include a large proportion of non-canonical tokens, it is definitely worth applying normalization in combination with embeddings.

Our investigation was limited by the amount of available training data. Adding data via self-training did not help. We observed mixed results for different types of a-priori filtering, but none of them yielded a steady improvement. A more promising direction might be post-selection, based on confidence scores or agreement among different taggers. Obviously another way to go is to add manually labeled data, some of which is available for more canonical domains. This would require a mapping of tagsets, and might be another good testbed to assess the contribution of normalization, which we leave for future work.

All code and distributable data used in this paper are available at `https://github.com/bplank/wnut-2017-pos-norm`.

Acknowledgments

We want to thank Héctor Martínez Alonso and Gertjan van Noord for valuable comments on earlier drafts of this paper. We are also grateful to the anonymous reviewers. This research has been supported by the Nuance Foundation and the University of Groningen High Performance Computing center.

References

Steven Abney. 2007. *Semisupervised learning for computational linguistics*. CRC Press.

Timothy Baldwin, Marie-Catherine de Marneffe, Bo Han, Young-Bum Kim, Alan Ritter, and Wei Xu. 2015. Shared tasks of the 2015 workshop on noisy user-generated text: Twitter lexical normalization and named entity recognition. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 126–135, Beijing, China. Association for Computational Linguistics.

Michael Beißwenger, Sabine Bartsch, Stefan Evert, and Kay-Michael Würzner. 2016. Empirist 2015: A shared task on the automatic linguistic annotation of computer-mediated communication and web corpora. In *Proceedings of the 10th Web as Corpus Workshop (WAC-X) and the EmpiriST Shared Task. Berlin, Germany*, pages 44–56.

Marcel Bollmann. 2013. POS tagging for historical texts with sparse training data. In *Proceedings of the 7th Linguistic Annotation Workshop and Interoperability with Discourse, LAW-ID@ACL 2013, August 8-9, 2013, Sofia, Bulgaria*, pages 11–18. The Association for Computer Linguistics.

Cristina Bosco, Fabio Tamburini, Andrea Bolioli, and Alessandro Mazzei. 2016. Overview of the evalita 2016 part of speech on twitter for italian task. In *Proceedings of Third Italian Conference on Computational Linguistics (CLiC-it 2016) & Fifth Evaluation Campaign of Natural Language Processing and Speech Tools for Italian. Final Workshop (EVALITA 2016). Associazione Italiana di Linguistica Computazionale (AILC)*.

Leo Breiman. 2001. Random forests. *Machine learning*, 45(1):5–32.

Jacob Eisenstein. 2013. What to do about bad language on the internet. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 359–369, Atlanta, Georgia. Association for Computational Linguistics.

Jennifer Foster, Özlem Cetinoglu, Joachim Wagner, Joseph Le Roux, Joakim Nivre, Deirdre Hogan, and Josef van Genabith. 2011. From news to comment: Resources and benchmarks for parsing the language of web 2.0. In *Proceedings of the 5th International Joint Conference on Natural Language Processing (IJCNLP)*.

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 *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 42–47, Portland, Oregon, USA. Association for Computational Linguistics.

Rob van der Goot and Gertjan van Noord. 2017. Monoise: Modeling noise using a modular normalization system. *Computational Linguistics in the Netherlands Journal*, 7.

Bo Han and Timothy Baldwin. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*, pages 368–378, Portland, Oregon, USA. Association for Computational Linguistics.

Dirk Hovy, Barbara Plank, and Anders Søgaard. 2014. When pos data sets don't add up: Combatting sample bias. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation (LREC-2014)*, pages 4472–4475.

Dieuwke Hupkes and Rens Bod. 2016. Pos-tagging of historical dutch. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC 2016)*, Paris, France. European Language Resources Association (ELRA).

Chen Li and Yang Liu. 2014. Improving text normalization via unsupervised model and discriminative reranking. In *Proceedings of the ACL 2014 Student Research Workshop*, pages 86–93, Baltimore, Maryland, USA. Association for Computational Linguistics.

Chen Li and Yang Liu. 2015. Joint POS tagging and text normalization for informal text. In *Proceedings of the Twenty-Fourth International Joint Conference on Artificial Intelligence, IJCAI 2015, Buenos Aires, Argentina, July 25-31, 2015*, pages 1263–1269.

Wang Ling, Chris Dyer, Alan W Black, and Isabel Trancoso. 2015. Two/too simple adaptations of word2vec for syntax problems. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1299–1304, Denver, Colorado. Association for Computational Linguistics.

Nikola Ljubešić, Tomaž Erjavec, and Darja Fišer. 2017. Adapting a state-of-the-art tagger for south slavic languages to non-standard text. In *Proceedings of the 6th Workshop on Balto-Slavic Natural Language Processing*, pages 60–68, Valencia, Spain. Association for Computational Linguistics.

David McClosky. 2010. *Any domain parsing: automatic domain adaptation for natural language parsing*. Ph.D. thesis, Brown University.

Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*.

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 *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia. Association for Computational Linguistics.

Slav Petrov and Ryan McDonald. 2012. Overview of the 2012 shared task on parsing the web. In *Notes of the First Workshop on Syntactic Analysis of Non-Canonical Language (SANCL)*, volume 59.

Barbara Plank. 2016. What to do about non-standard (or non-canonical) language in NLP. In *KONVENS*.

Barbara Plank, Anders Søgaard, and Yoav Goldberg. 2016. Multilingual part-of-speech tagging with bidirectional long short-term memory models and auxiliary loss. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, pages 412–418, Berlin, Germany. Association for Computational Linguistics.

Alan Ritter, Sam Clark, Mausam, and Oren Etzioni. 2011. Named entity recognition in tweets: An experimental study. In *Proceedings of the 2011 Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534, Edinburgh, Scotland, UK. Association for Computational Linguistics.

Erik Tjong Kim Sang. 2016. Improving Part-of-Speech Tagging of Historical Text by First Translating to Modern Text. In *2nd IFIP International Workshop on Computational History and Data-Driven Humanities*. Springer Verlag.

A. Viterbi. 1973. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Trans. Inform. Theory*, 13(2):260–269.

Yi Yang and Jacob Eisenstein. 2016. Part-of-speech tagging for historical english. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1318–1328, San Diego, California. Association for Computational Linguistics.