Normalizing Social Media Texts by Combining Word Embeddings and Edit Distances in a Random Forest Regressor

Rob van der Goot
University of Groningen
R.van.der.Goot@rug.nl

Abstract

In this work, we adapt the traditional framework for spelling correction to the more novel task of normalization of social media content. To generate possible normalization candidates, we complement the traditional approach with a word embeddings model. To rank the candidates we will use a random forest regressor, combining the features from the generation with some N-gram features. The N-gram model contributes significantly to the model, because no other features account for short-distance relations between words. A random forest regressor fits this task very well, presumably because it can model the different types of corrections. Additionally we show that 500 annotated sentences should be enough training data to train this system reasonably well on a new domain. Our proposed system performs slightly worse compared to the state-of-the-art. The main advantage is the simplicity of the model, allowing for easy expansions.

Keywords: Normalization, Noise, Word Embeddings, Random Forest

1. Introduction

Because the task of normalization has a lot of similarities with the task of spelling correction, many of the same methods can be used. The standard framework for spelling correction consists of three steps: error detection, candidate generation and candidate ranking. In this paper, we will use this framework, but skip the step of error detection; because this model is meant to be used in a pipeline, this task can be postponed, so that a more informed decision can be made for this crucial step. Traditionally, the steps in this framework were based on a combination of lexical and phonetic distance measures. This approach was focused on spelling correction and motivated by the fact that every word that needs correction is a spelling error or typographical error. These alternations occur a lot in social media data, but are complemented by other types of alternations which are more domain specific. These include slang, abbreviations, domain-specific conventions and new linguistic structures.

In this work, we will expand a traditional spelling correction method to adapt to the noisy social media domain. Word embeddings are exploited to complement the traditional candidate generation which is based on lexical and phonetical distances. Furthermore, a random forest regressor will be used to combine the features which are mainly collected during the generation. This simple system allows for easy expansions, for example: multiword replacements, word deletion or word insertion. Evaluation will be done on the standard benchmark for normalization of English social media texts: LexNorm 1.2 (Yang and Eisenstein, 2013). An example sentence from this dataset is shown in Sentence 1.

(1) new pix comming tomoroe
new pictures coming tomorrow

2. Related Work

Han and Baldwin (2011) describe one of the first normalization approaches tailored for the social media domain. First, candidates are generated by finding lexically and phonetically close words. Ranking is then done with a support vector machine. A wide range of features is used: dependency tree distance, lexical edit distance, phonetic edit distance, prefix substring, suffix substring and the longest common substring.

A completely different approach is taken by Hassan and Menezes (2013). Here, a bipartite graph is used with on one side the words, and on the other side n-gram contexts in which these words occur. In this bipartite graph, Markov Random Walks are used to generate correction candidates. Ranking is done afterwards, based on a lexical similarity distance.

Xu et al. (2015) use lexical and phonetic features on the syllable level instead of the word or character level. Syllables are extracted from erroneous words and are converted to an ARPAbet representation (Rabiner and Juang, 1993). The ARPAbet encoding of the erroneous token can be compared to ARPAbet encodings of words taken from a dictionary. Edit distances on the ARPAbet encoding are then used to compare possible candidates.

An ensemble reranking method is proposed by Li and Liu (2014), where four different systems for normalization are combined including a spell checker and some machine translation methods. Building further on this work, Li and Liu (2015) created a joint model for normalization and POS tagging. The candidate lists of the reranking model discussed in the previous paragraph are used in a Viterbi decoding (Viterbi, 1973). Traditionally, all possible POS tags for a word in the sentence are used in the encoding, but in the new model all possible POS tags for all possible corrections are used in the encoding. This model achieves state-of-the art performance on the LexNorm dataset as well as on the standard benchmark for POS tagging of Twitter data (Owoputi et al., 2013).

3. Method

Our system is based on two steps: candidate generation and the ranking of the candidates. Both of them are discussed in more detail below.

3.1. Candidate Generation

Candidate generation for unintended disfluencies is a much studied problem; most approaches make use of the lexical or phonetic properties of a word to find similar words in a vocabulary. Due to the vast amount of work, and the good results on the task of finding lexically similar words, we consider this task to be as good as solved. For this reason, we will use the Aspell spell checker for this task, which achieves a recall of 98% on a list of common misspellings[1]. It uses a lexical edit distance combined with a phonetic edit distance based on the Double Metaphone algorithm (Philips, 2000). Aspell is slightly modified to be able to process words consisting of only one character and we include phonetic information about numerals.

To find normalizations replacements for intended noise, we need a more meaning-driven approach. Word embeddings capture the meaning of a word by using the context it occurs in. A big advantage of this method is the fact that word embeddings are trained on huge amounts of unlabeled data, which is readily available for the social media domain. Words that occur in similar contexts will be close to each other in the vector space, and are thus good normalization candidates. We will use the word embeddings model of Godin et al. (2015), which is originally used for named entity recognition for Tweets. This skip-gram model is trained on 400 million Tweets, uses 400 dimensions, and contains over 3 million types.

3.2. Candidate Ranking

The task of finding the correct candidate can be interpreted as a binary classification task as there are only two classes we are trying to distinguish: correct and incorrect. This interpretation enables the use of a binary classifier, but also introduces some problems. Firstly, a binary classifier can never guarantee to only assign one instance to a class, let alone a list of possible candidates. This is solved by ordering the candidates using the probabilities of being in the 'correct' class.

Secondly, the training of a classifier with very few instances in one class is a problem. Empirical experiments on our training data shows that 85% of all tokens should be left untouched, so simple ranking on one binary feature, and thus zeroing out the others, results in an accuracy above 85%. This is solved by removing the original word before the training of the classifier. Because the original word should often stay untouched, it is always used as the highest ranked candidate in the candidate list. Following from this, the ranking is only evaluated on the erroneous tokens, so only these tokens will be used as training data.

A Random Forest regression model is chosen because its structure can adjust well to the underlying problem. This model combines multiple decision trees, trained on random subsets of the training data. Each input will follow a path down in every decision tree resulting in a prediction value for each tree. These values are then averaged, which results in one final prediction. This model is very suitable, because the underlying problem is not binary; we are trying to normalize different types of disfluencies, which might have very different values for the different features. More concretely, this model can learn that a high value only on feature A can be enough to classify it as the correct candidate, without excluding that feature B can have the same effect. We use the Random Forest implementation of Scikit-Learn (Pedregosa et al., 2011) with its default parameters, except for the number of estimators, which is set to 100.

The following features are used to train the random forest model:

- A score used by Aspell to indicate the lexical and phonetical edit distance and binary features indicating if the candidate and the original word can be found in the Aspell dictionary.
- The distance in the vector space of the word embeddings model between the original word and the correction candidate.
- Uni- and bi- gram probabilities, taken from two different n-gram models: a noisy twitter n-gram model (Herdağdelen, 2013), and a model based on clean texts (Brants and Franz, 2006).

4. Evaluation

4.1. Data

Two different normalization datasets are used in this work:

- Train set: 2,577 Tweets annotated with normalization (Li and Liu, 2014). This dataset consists of tweets taken from the Edinburgh Twitter Corpus (Petrović et al., 2010), and are annotated using Amazon Turk.
- Test set: The LexNorm dataset (Han and Baldwin, 2011), 549 tweets from a different period annotated by different annotators.

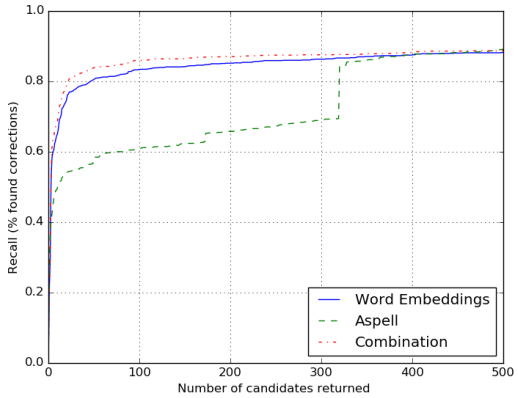
Both of these datasets use pre-tokenized tweets and only allow corrections on the word level. This setup ensures that our testing is robust with respect to biases in the annotation style and time period.

4.2. Generation

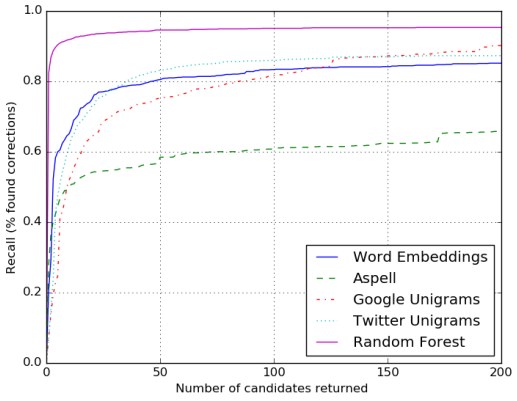
The generation is evaluated only on the words that are corrected in the annotated data. Two methods are compared, the traditional Aspell, and the generation from the word embeddings. However, our main interest is how well they can complement each other. For this reason, we included a naive combinatory method; this method simply takes equal numbers of candidates from the other 2 methods.

The individual and combined results are shown in Figure 1a. Word embeddings work better for this task than the traditional Aspell methods and combining them with a simple combination method already proves that they can complement each other. Additionally, we can see that the improvement in recall using a list of more than 100 candidates is moderate. One exception is the recall improvement for Aspell at 318 candidates. This is because at the Aspell candidate list for the common token 'u', the correct word

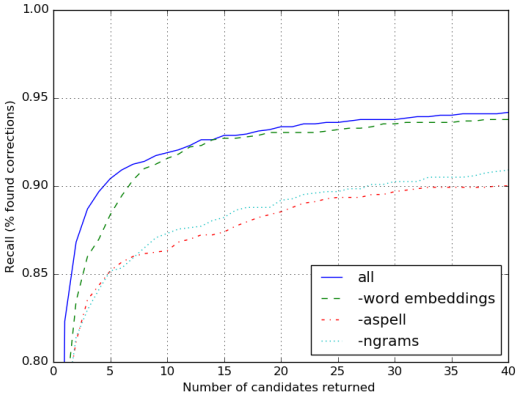
[1] http://aspell.net/test/orig/



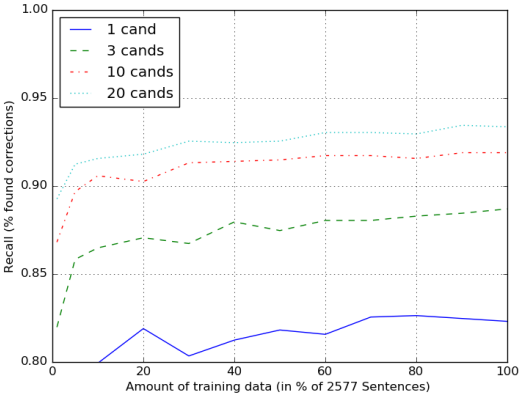
(a) Results for candidate generation



(b) Comparison of ranking on single features



(c) Results of ablation experiments



(d) Effect of quantity of training data

Figure 1: Evaluation results of different experiments

'you' is at the 318th position. This word has already been found by the word embeddings (at position 3), so it does not affect the combination line. This one word accounts for a big part of the performance difference.

4.3. Ranking

Candidate ranking of a normalization system is usually only evaluated on the highest scoring candidate for each erroneous word. We will focus on two aspects: a high recall combined with a low number of candidates. A high recall ensures that the right candidate is in the list so that the following application has access to it and a low number of candidates is important for efficiency down the pipeline.

Table 1 shows a comparison of our system with the previous work that reports scores for different numbers of candidates, as well as the best performing system for this task. Note that our generation is slightly worse compared to the previous systems, this can probably be improved by adding some domain specific heuristics or by tweaking the word embeddings model. However, the ranking works surprisingly well, after only 20 candidates the upperbound is reached. The state-of-the-art system performs better on the top 1 candidate, but this system is a lot more complex. Unfortunately, the results for other numbers of candidates are not reported.

4.4. Feature Importance

First, we will evaluate how our features can perform on their own, then we will see how important the feature are with respect to the model in an ablation experiment.

Figure 1b shows the results of ranking on single features. Word embeddings have the best performance for a low number of candidates while the Twitter unigrams have the highest performance with more candidates. Additionally, we can see that Twitter N-grams generally work better, even though the google N-gram model is based on clean data. Presumably, this is because the Twitter N-grams have less sparsity with respect to the test data.

The ablation experiments are done on feature-groups. Grouping is done based on source level. The same grouping as in Section 3.2. is used. The results of the ablation experiments are shown in Figure 1c. Surprisingly, the word embeddings are the least important for ranking. This is

System	top1	top3	top10	top20	upper bound
Li and Liu (2012)	73.0	81.9	86.7	89.2	94.2
Li and Liu (2014)	77.14	86.96	93.04	94.82	95.90
Li and Liu (2015)	87.58				
Our system	82.31	88.70	91.89	93.37	93.37

Table 1: Recall of our system compared to previous work

probably because they reflect information from only one perspective, the distance in the word vectors, whereas the N-grams reflect on unigrams and bigrams from two different language models. Furthermore, the N-grams reflect on the relations of close words, which are important for grammatical correctness. Aspell appears to be very important for the ranking step, presumably because most types of alternations use some sort of lexical or phonetic variation of the intended word.

4.5. Reduce Training Data

To adapt this model to another domain, three resources are needed. A word embeddings model, an N-gram model and annotated data. Because an annotated dataset is the most expensive resource to acquire, only this resource is tested for quantity. Figure 1d shows how the performance drops when we decrease the amount of training data. After using 20% ($\approx$ 500 sentences) of the training data the improvements in performance are quite small.

5. Conclusion

We have shown that a spelling correction system can be converted to a normalization system by using modern techniques. Word embeddings can complement the lexical and phonetical approaches well for candidate generation because it targets other types of noise. Additionally, a random forest regressor can fit well to the normalization task, presumably because it can model the different kinds of noise in different parts of its trees. There are still plenty of improvements possible for this system, Aspell is not designed for this domain and the word embeddings model was preprocessed for another task. Almost no parameters have been tuned for the random forest regressor. Another source of improvement could be the addition of features.
Further directions include the addition of multiword replacements, but mainly the use of this system in a pipeline. Only then its usefulness can be properly tested. Our system outputs the whole candidate lists, and is made available on the authors website.

Acknowledgements I would like to thank Gertjan van Noord, my other colleagues and the anonymous reviewers for their valuable feedback, and the Nuance Foundation for funding the Parsing Algorithms for Uncertain Input project.

6. Bibliographical References

Brants, T. and Franz, A. (2006). Web 1T 5-gram version 1. Technical report, Google.

Godin, F., Vandersmissen, B., De Neve, W., and Van de Walle, R. (2015). Multimedia Lab @ ACL WNUT NER shared task: Named entity recognition for Twitter microposts using distributed word representations. In *Proceedings of the Workshop on Noisy User-generated Text*, pages 146–153, Beijing, China, July. Association for Computational Linguistics.

Han, B. and Baldwin, T. (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, June. Association for Computational Linguistics.

Hassan, H. and Menezes, A. (2013). Social text normalization using contextual graph random walks. In *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1577–1586, Sofia, Bulgaria, August. Association for Computational Linguistics.

Herdağdelen, A. (2013). Twitter n-gram corpus with demographic metadata. *Language resources and evaluation*, 47(4):1127–1147.

Li, C. and Liu, Y. (2012). Improving text normalization using character-blocks based models and system combination. In *Proceedings of COLING 2012*, pages 1587–1602, Mumbai, India, December.

Li, C. and Liu, Y. (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, June. Association for Computational Linguistics.

Li, C. and Liu, Y. (2015). Joint pos tagging and text normalization for informal text. In *Proceedings of IJCAI*.

Owoputi, O., O'Connor, B., Dyer, C., Gimpel, K., Schneider, N., and Smith, N. A. (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, June. Association for Computational Linguistics.

Pedregosa, F., Varoquaux, G., Gramfort, A., Michel, V., Thirion, B., Grisel, O., Blondel, M., Prettenhofer, P., Weiss, R., Dubourg, V., et al. (2011). Scikit-learn: Machine learning in python. *The Journal of Machine Learning Research*, 12:2825–2830.

Petrović, S., Osborne, M., and Lavrenko, V. (2010). The Edinburgh Twitter corpus. In *Proceedings of the NAACL HLT 2010 Workshop on Computational Linguistics in a World of Social Media*, pages 25–26, Los Angeles, California, USA, June. Association for Computational Linguistics.

Philips, L. (2000). The double metaphone search algorithm. In *C/C++ users journal*, volume 18, pages 38–43.

Rabiner, L. and Juang, B.-H. (1993). Fundamentals of speech recognition. Technical report, Prentice hall.

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

Xu, K., Xia, Y., and Lee, C.-H. (2015). Tweet normalization with syllables. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 920–928, Beijing, China, July. Association for Computational Linguistics.

Yang, Y. and Eisenstein, J. (2013). A log-linear model for unsupervised text normalization. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 61–72, Seattle, Washington, USA, October. Association for Computational Linguistics.