Normalizing tweets with edit scripts and recurrent neural embeddings

Grzegorz Chrupała
Tilburg Center for Cognition and Communication
Tilburg University
g.chrupala@uvt.nl

Abstract

Tweets often contain a large proportion of abbreviations, alternative spellings, novel words and other non-canonical language. These features are problematic for standard language analysis tools and it can be desirable to convert them to canonical form. We propose a novel text normalization model based on learning edit operations from labeled data while incorporating features induced from unlabeled data via character-level neural text embeddings. The text embeddings are generated using an Simple Recurrent Network. We find that enriching the feature set with text embeddings substantially lowers word error rates on an English tweet normalization dataset. Our model improves on state-of-the-art with little training data and without any lexical resources.

1 Introduction

A stream of posts from Twitter contains text written in a large variety of languages and writing systems, in registers ranging from formal to internet slang. Substantial effort has been expended in recent years to adapt standard NLP processing pipelines to be able to deal with such content. One approach has been text normalization, i.e. transforming tweet text into a more canonical form which standard NLP tools expect. A multitude of resources and approaches have been used to deal with normalization: handcrafted and (semi-)automatically induced dictionaries, language models, finite state transducers, machine translation models and combinations thereof. Methods such as those of Han and Baldwin (2011), Liu et al. (2011), Gouws et al. (2011) or Han et al. (2012) are unsupervised but they typically use many adjustable parameters which need to be tuned on some annotated data. In this work we suggest a simple, supervised character-level string transduction model which easily incorporates features automatically learned from large amounts of unlabeled data and needs only a limited amount of labeled training data and no lexical resources.

Our model learns sequences of edit operations from labeled data using a Conditional Random Field (Lafferty et al., 2001). Unlabeled data is incorporated following recent work on using character-level text embeddings for text segmentation (Chrupała, 2013), and word and sentence boundary detection (Evang et al., 2013). We train a recurrent neural network language model (Mikolov et al., 2010; Mikolov, 2012b) on a large collection of tweets. When run on new strings, the activations of the units in the hidden layer at each position in the string are recorded and used as features for training the string transduction model.

The principal contributions of our work are: (i) we show that a discriminative sequence labeling model is apt for text normalization and performs at state-of-the-art levels with small amounts of labeled training data; (ii) we show that character-level neural text embeddings can be used to effectively incorporate information from unlabeled data into the model and can substantially boost text normalization performance.

2 Methods

Many approaches to text normalization adopt the noisy channel setting, where the model normalizing source string s into target canonical form t is factored into two parts: $\hat{t} = \arg\max_t P(t)P(s|t)$. The error term $P(s|t)$ models how canonical strings are transformed into variants such as e.g. misspellings, emphatic lengthenings or abbreviations. The language model $P(t)$ encodes which target strings are probable.

We think this decomposition is less appropriate

Input	c	␣	w	a	t
Edit	DEL	INS(see)	NIL	INS(h)	NIL
Output		see␣	w	ha	t

Table 1: Example edit script.

in the context of text normalization than in applications from which it was borrowed such as Machine Translations. This is because it is not obvious what kind of data can be used to estimate the language model: there is plentiful text from the source domain, but little of it is in normalized *target* form. There is also much edited text such as news text, but it comes from a very different domain. One of the main advantages of the noisy channel decomposition is that is makes it easy to exploit large amounts of unlabeled data in the form of a language model. This advantage does not hold for text normalization.

We thus propose an alternative approach where normalization is modeled directly, and which enables easy incorporation of unlabeled data from the *source* domain.

2.1 Learning to transduce strings

Our string transduction model works by learning the sequence of edits which transform the input string into the output string. Given a pair of strings such a sequence of edits (known as the shortest edit script) can be found using the DIFF algorithm (Miller and Myers, 1985; Myers, 1986). Our version of DIFF uses the following types of edits:

- NIL – no edits,
- DEL – delete character at this position,
- INS$(\cdot)$ – insert specified string before character at this position.[1]

Table 1 shows a shortest edit script for the pair of strings (*c wat, see what*).

We use a sequence labeling model to learn to label input strings with edit scripts. The training data for the model is generated by computing shortest edit scripts for pairs of original and normalized strings. As a sequence labeler we use Conditional Random Fields (Lafferty et al., 2001). Once trained the model is used to label new strings and the predicted edit script is applied to the input string producing the normalized output string. Given source string s the predicted target string $\hat{t}$ is:

$$\hat{t} = \arg\max_t P(\text{ses}(s,t)|s)$$

where $e = \text{ses}(s,t)$ is the shortest edit script mapping s to t. $P(e|s)$ is modeled with a linear-chain Conditional Random Field.

2.2 Character-level text embeddings

Simple Recurrent Networks (SRNs) were introduced by Elman (1990) as models of temporal, or sequential, structure in data, including linguistic data (Elman, 1991). More recently SRNs were used as language models for speech recognition and shown to outperform classical n-gram language models (Mikolov et al., 2010; Mikolov, 2012b). Another version of recurrent neural nets has been used to generate plausible text with a character-level language model (Sutskever et al., 2011). We use SRNs to induce character-level text representations from unlabeled Twitter data to use as features in the string transduction model.

The units in the hidden layer at time t receive connections from input units at time t and also from the hidden units at the previous time step $t-1$. The hidden layer predicts the state of the output units at the next time step $t+1$. The input vector $w(t)$ represents the input element at current time step, here the current character. The output vector $y(t)$ represents the predicted probabilities for the next character. The activation s_j of a hidden unit j is a function of the current input and the state of the hidden layer at the previous time step: $t-1$:

$$s_j(t) = \sigma\left(\sum_{i=1}^{I} w_i(t)U_{ji} + \sum_{l=1}^{L} s_j(t-1)W_{jl}\right)$$

where σ is the sigmoid function and U_{ji} is the weight between input component i and hidden unit j, while W_{jl} is the weight between hidden unit l at time $t-1$ and hidden unit j at time t. The representation of recent history is stored in a limited number of recurrently connected hidden units. This forces the network to make the representation compressed and abstract rather than just memorize literal history. Chrupała (2013) and Evang et al. (2013) show that these text embeddings can be useful as features in textual segmentation tasks. We use them to bring in information from unlabeled data into our string transduction model and then train a character-level SRN language model on unlabeled tweets. We run the trained model on

[1] The input string is extended with an empty symbol to account for the cases where an insertion is needed at the end of the string.

```
@YuszLAL100A 暇すぎるｗｗｗｗとか麺役者についてる…（ゝ ＞
晒せ 信じに行けていいんだな…。 RT @yaepdrrafa:
@fsch_chany siaaa,, dobek taha subus sama kiri kabur
wanak... hahah
なかなかない。
やばい
But I'm the good first-Good Chulc
```

Figure 1: Tweets randomly generated with an SRN

new tweets and record the activation of the hidden layer at each position as the model predicts the next character. These activation vectors form our text embeddings: they are discretized and used as input features to the supervised sequence labeler as described in Section 3.4.

3 Experimental Setup

We limit the size of the string alphabet by always working with UTF-8 encoded strings, and using bytes rather than characters as basic units.

3.1 Unlabeled tweets

In order to train our SRN language model we collected a set of tweets using the Twitter sampling API. We use the raw sample directly without filtering it in any way, relying on the SRN to learn the structure of the data. The sample consists of 414 million bytes of UTF-8 encoded in a variety of languages and scripts text. We trained a 400-hidden-unit SRN, to predict the next byte in the sequence using backpropagation through time. Input bytes were encoded using one-hot representation. We modified the RNNLM toolkit (Mikolov, 2012a) to record the activations of the hidden layer and ran it with the default learning rate schedule. Given that training SRNs on large amounts of text takes a considerable amount of time we did not vary the size of the hidden layer. We did try to filter tweets by language and create specific embeddings for English but this had negligible effect on tweet normalization performance.

The trained SRN language model can be used to generate random text by sampling the next byte from its predictive distribution and extending the string with the result. Figure 1 shows example strings generated in this way: the network seems to prefer to output pseudo-tweets written consistently in a single script with words and pseudo-words mostly from a single language. The generated byte sequences are valid UTF-8 strings.

In Table 2 in the first column we show the suffix of a string for which the SRN is predicting the last byte. The rest of each row shows the nearest neighbors of this string in embedding space, i.e.

should h	should d	will s	will m	should a
@justth	@neenu	@raven_	@lanae	@despic
maybe	u maybe y	cause i	wen i	when i

Table 2: Nearest neighbors in embedding space.

strings for which the SRN is activated in a similar way when predicting its last byte as measured by cosine similarity.

3.2 Normalization datasets

A difficulty in comparing approaches to tweet normalization is the sparsity of publicly available datasets. Many authors evaluate on private tweet collections and/or on the text message corpus of Choudhury et al. (2007).

For English, Han and Baldwin (2011) created a small tweet dataset annotated with normalized variants at the word level. It is hard to interpret the results from Han and Baldwin (2011), as the evaluation is carried out by assuming that the words to be normalized are known in advance: Han et al. (2012) remedy this shortcoming by evaluating a number of systems without prespecifying ill-formed tokens. Another limitation is that only word-level normalization is covered in the annotation; e.g. splitting or merging of words is not allowed. The dataset is also rather small: 549 tweets, which contain 2139 annotated out-of-vocabulary (OOV) words. Nevertheless, we use it here for training and evaluating our model. This dataset does not specify a development/test split. In order to maximize the size of the training data while avoiding tuning on test data we use a split cross-validation setup: we generate 10 cross-validation folds, and use 5 of them during development to evaluate variants of our model. The best performing configuration is then evaluated on the remaining 5 cross-validation folds.

3.3 Model versions

The simplest way to normalize tweets with a string transduction model is to treat whole tweets as input sequences. Many other tweet normalization methods work in a word-wise fashion: they first identify OOV words and then replace them with normalized forms. Consequently, publicly available normalization datasets are annotated at word level. We can emulate this setup by training the sequence labeler on words, instead of whole tweets. This approach sacrifices some generality, since transformations involving multiple words cannot

be learned. However, word-wise models are more comparable with previous work. We investigated the following models:

- OOV-ONLY is trained on individual words and in-vocabulary (IV) words are discarded for training, and left unchanged for prediction.[2]
- ALL-WORDS is trained on all words and allowed to change IV words.
- DOCUMENT is trained on whole tweets.

Model OOV-ONLY exploits the setting when the task is constrained to only normalize words absent from a reference dictionary, while DOCUMENT is the one most generally applicable but does not benefit from any constraints. To keep model size within manageable limits we reduced the label set for models ALL-WORDS and DOCUMENT by replacing labels which occur less than twice in the training data with NIL. For OOV-ONLY we were able to use the full label set. As our sequence labeling model we use the Wapiti implementation of Conditional Random Fields (Lavergne et al., 2010) with the L-BFGS optimizer and elastic net regularization with default settings.

3.4 Features

We run experiments with two feature sets: N-GRAM and N-GRAM+SRN. N-GRAM are character n-grams of size 1–3 in a window of $(-2, +2)$ around the current position. For the N-GRAM+SRN feature set we augment N-GRAM with features derived from the activations of the hidden units as the SRN is trying to predict the current character. In order to use the activations in the CRF model we discretize them as follows. For each of the $K = 10$ most active units out of total $J = 400$ hidden units, we create features $(f(1) \ldots f(K))$ defined as $f(k) = 1$ if $s_{j(k)} > 0.5$ and $f(k) = 0$ otherwise, where $s_{j(k)}$ returns the activation of the k^{th} most active unit.

3.5 Evaluation metrics

As our evaluation metric we use word error rate (WER) which is defined as the Levenshtein edit distance between the predicted word sequence $\hat{t}$ and the target word sequence t, normalized by the total number of words in the target string. A more generally applicable metric would be character error rate, but we report WERs to make our results easily comparable with previous work. Since the English dataset is pre-tokenized and only covers word-to-word transformations, this choice has little importance here and character error rates show a similar pattern to word error rates.

[2] We used the IV/OOV annotations in the Han et al. (2012) dataset, which are automatically derived from the aspell dictionary.

Model	Features	WER (%)
NO-OP		11.7
DOCUMENT	NGRAM	6.8
DOCUMENT	NGRAM+SRN	5.7
ALL WORDS	NGRAM	7.2
ALL WORDS	NGRAM+SRN	5.0
OOV-ONLY	NGRAM	5.1
OOV-ONLY	NGRAM+SRN	**4.5**

Table 3: WERs on development data.

9 cont continued	5 gon gonna
4 bro brother	4 congrats congratulations
3 yall you	3 pic picture
2 wuz what's	2 mins minutes
2 juss just	2 fb facebook

Table 4: Improvements from SRN features.

4 Results

Table 3 shows the results of our development experiments. NO-OP is a baseline which leaves text unchanged. As expected the most constrained model OOV-ONLY outperforms the more generic models on this dataset. For all model variations, adding SRN features substantially improves performance: the relative error reductions range from 12% for OOV-ONLY to 30% for ALL-WORDS. Table 4 shows the non-unique normalizations made by the OOV-ONLY model with SRN features which were missed without them. SRN features seem to be especially useful for learning long-range, multi-character edits, e.g. *fb* for *facebook*.

Table 5 shows the non-unique normalizations which were missed by the best model: they are a mixture of relatively standard variations which happen to be infrequent in our data, like *tonite* or *gf*, and a few idiosyncratic respellings like *uu* or *bhee*. Our supervised approach makes it easy to address the first type of failure by simply annotating additional training examples.

Table 6 presents evaluation results of several approaches reported in Han et al. (2012) as well as the model which did best in our development experiments. HB-dict is the Internet slang dictionary from Han and Baldwin (2011). GHM-dict is the automatically constructed dictionary from

4 1 one	2 withh with
2 uu you	2 tonite tonight
2 thx thanks	2 thiis this
2 smh somehow	2 outta out
2 n in	2 m am
2 hmwrk homework	2 gf girlfriend
2 fxckin fucking	2 dha the
2 de the	2 d the
2 bhee be	2 bb baby

Table 5: Missed transformations.

Method	WER (%)
NO-OP	11.2
HB-dict	6.6
GHM-dict	7.6
S-dict	9.7
Dict-combo	4.9
Dict-combo+HB-norm	7.9
OOV-ONLY NGRAM+SRN (test)	**4.8**

Table 6: WERs compared to previous work.

Gouws et al. (2011); S-dict is the automatically constructed dictionary from (Han et al., 2012); Dict-combo are all the dictionaries combined and Dict-combo+HB-norm are all dictionaries combined with approach of Han and Baldwin (2011). The WER reported for OOV-ONLY NGRAM+SRN is on the test folds only. The score on the full dataset is a bit better: 4.66%. As can be seen our approach it the best performing approach overall and in particular it does much better than all of the single dictionary-based methods. Only the combination of all the dictionaries comes close in performance.

5 Related work

In the field of tweet normalization the approach of Liu et al. (2011, 2012) shows some similarities to ours: they gather a collection of OOV words together with their canonical forms from the web and train a character-level CRF sequence labeler on the edit sequences computed from these pairs. They use this as the error model in a noisy-channel setup combined with a unigram language model. In addition to character n-gram features they use phoneme and syllable features, while we rely on the SRN embeddings to provide generalized representations of input strings.

Kaufmann and Kalita (2010) trained a phrase-based statistical translation model on a parallel text message corpus and applied it to tweet normalization. In comparison to our first-order linear-chain CRF, an MT model with reordering is more flexible but for this reason needs more training data. It also suffers from language model mismatch mentioned in Section 2: optimal results were obtained by using a low weight for the language model trained on a balanced text corpus.

Many other approaches to tweet normalization are more unsupervised in nature (e.g. Han and Baldwin, 2011; Gouws et al., 2011; Xue et al., 2011; Han et al., 2012). They still require annotated development data for tuning parameters and a variety of heuristics. Our approach works well with similar-sized training data, and unlike unsupervised approaches can easily benefit from more if it becomes available. Further afield, our work has connections to research on morphological analysis: for example Chrupała et al. (2008) use edit scripts to learn lemmatization rules while Dreyer et al. (2008) propose a discriminative model for string transductions and apply it to morphological tasks. While Chrupała (2013) and Evang et al. (2013) use character-level SRN text embeddings for learning segmentation, and recurrent nets themselves have been used for sequence transduction (Graves, 2012), to our knowledge *neural text embeddings* have not been previously applied to string transduction.

6 Conclusion

Learning sequences of edit operations from examples while incorporating unlabeled data via neural text embeddings constitutes a compelling approach to tweet normalization. Our results are especially interesting considering that we trained on only a small annotated data set and did not use any other manually created resources such as dictionaries. We want to push performance further by expanding the training data and incorporating existing lexical resources. It will also be important to check how our method generalizes to other language and datasets (e.g. de Clercq et al., 2013; Alegria et al., 2013).

The general form of our model can be used in settings where normalization is not limited to word-to-word transformations. We are planning to find or create data with such characteristics and evaluate our approach under these conditions.

References

Iñaki Alegria, Nora Aranberri, Víctor Fresno, Pablo Gamallo, Lluis Padró, Iñaki San Vicente, Jordi Turmo, and Arkaitz Zubiaga. 2013. Introducción a la tarea compartida Tweet-Norm 2013: Normalización léxica de tuits en español. In *Workshop on Tweet Normalization at SEPLN (Tweet-Norm)*, pages 36–45.

Monojit Choudhury, Rahul Saraf, Vijit Jain, Animesh Mukherjee, Sudeshna Sarkar, and Anupam Basu. 2007. Investigation and modeling of the structure of texting language. *International Journal of Document Analysis and Recognition (IJDAR)*, 10(3-4):157–174.

Grzegorz Chrupała. 2013. Text segmentation with character-level text embeddings. In *ICML Workshop on Deep Learning for Audio, Speech and Language Processing*.

Grzegorz Chrupała, Georgiana Dinu, and Josef Van Genabith. 2008. Learning morphology with Morfette. In *Proceedings of the 6th edition of the Language Resources and Evaluation Conference*.

Orphée de Clercq, Bart Desmet, Sarah Schulz, Els Lefever, and Véronique Hoste. 2013. Normalization of Dutch user-generated content. In *9th International Conference on Recent Advances in Natural Language Processing (RANLP-2013)*, pages 179–188. INCOMA Ltd.

Markus Dreyer, Jason R Smith, and Jason Eisner. 2008. Latent-variable modeling of string transductions with finite-state methods. In *Proceedings of the conference on empirical methods in natural language processing*, pages 1080–1089. Association for Computational Linguistics.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*, 14(2):179–211.

Jeffrey L Elman. 1991. Distributed representations, simple recurrent networks, and grammatical structure. *Machine learning*, 7(2-3):195–225.

Kilian Evang, Valerio Basile, Grzegorz Chrupała, and Johan Bos. 2013. Elephant: Sequence labeling for word and sentence segmentation. In *Empirical Methods in Natural Language Processing*.

Stephan Gouws, Dirk Hovy, and Donald Metzler. 2011. Unsupervised mining of lexical variants from noisy text. In *Proceedings of the First workshop on Unsupervised Learning in NLP*, pages 82–90. Association for Computational Linguistics.

Alex Graves. 2012. Sequence transduction with recurrent neural networks. arXiv:1211.3711.

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-Volume 1*, pages 368–378. Association for Computational Linguistics.

Bo Han, Paul Cook, and Timothy Baldwin. 2012. Automatically constructing a normalisation dictionary for microblogs. In *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning*, pages 421–432. Association for Computational Linguistics.

Max Kaufmann and Jugal Kalita. 2010. Syntactic normalization of Twitter messages. In *International conference on natural language processing, Kharagpur, India*.

John D. Lafferty, Andrew McCallum, and Fernando C. N. Pereira. 2001. Conditional Random Fields: Probabilistic models for segmenting and labeling sequence data. In *Proceedings of the Eighteenth International Conference on Machine Learning*, ICML '01, pages 282–289. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA.

Thomas Lavergne, Olivier Cappé, and François Yvon. 2010. Practical very large scale CRFs. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pages 504–513. Association for Computational Linguistics.

Fei Liu, Fuliang Weng, and Xiao Jiang. 2012. A broad-coverage normalization system for social media language. In *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics: Long Papers-Volume 1*, pages 1035–1044. Association for Computational Linguistics.

Fei Liu, Fuliang Weng, Bingqing Wang, and Yang Liu. 2011. Insertion, deletion, or substitution?: normalizing text messages without pre-categorization nor supervision. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies: short papers-Volume 2*, pages 71–76. Association for Computational Linguistics.

Tomáš Mikolov. 2012a. Recurrent neural network language models. http://rnnlm.org.

Tomáš Mikolov. 2012b. *Statistical language models based on neural networks*. Ph.D. thesis, Brno University of Technology.

Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černocký, and Sanjeev Khudanpur. 2010. Recurrent neural network based language model. In *Interspeech*, pages 1045–1048.

Webb Miller and Eugene W Myers. 1985. A file comparison program. *Software: Practice and Experience*, 15(11):1025–1040.

Eugene W Myers. 1986. An O(ND) difference algorithm and its variations. *Algorithmica*, 1(1-4):251–266.

Ilya Sutskever, James Martens, and Geoffrey E Hinton. 2011. Generating text with recurrent neural networks. In *Proceedings of the 28th International Conference on Machine Learning (ICML-11)*, pages 1017–1024.

Zhenzhen Xue, Dawei Yin, and Brian D Davison. 2011. Normalizing microtext. In *Proceedings of the AAAI-11 Workshop on Analyzing Microtext*, pages 74–79.