Normalization of Social Media Text using Deep Neural Networks

Ajay Shankar Tiwari
Jadavpur University
India
ajaytiwari0210@hotmail.com

Sudip Kumar Naskar
Jadavpur University
India
sudip.naskar@cse.jdvu.ac.in

Abstract

This paper sets out to investigate ways of normalizing noisy text that appear on social media platforms like Facebook, Twitter, Whatsapp, etc. We proposed a deep learning based approach to text normalization using Recurrent Neural Network (RNN) based Encoder–Decoder architecture with Long Short Term Memory (LSTM). To circumvent the unavailability of suitable large noisy–clean parallel dataset, we developed synthetic datasets. We trained and evaluated the proposed model on our synthetic datasets and the WNUT[1] shared task dataset. The uniqueness of our approach is in the use of synthetic datasets in a transfer learning approach for improving the performance of text normalization based on deep neural models. Our transfer learning based deep neural model produced state-of-the-art results (F1 score 0.9098) outperforming the previous best performing model on the WNUT test set by 7%.

1 Introduction

There is a large quantity of user-generated content on the web, characterized by social media, creativity and individuality, which has created problems at two levels. Firstly, social media text is often unsuitable for various Natural Language Processing (NLP) tasks, such as Information Retrieval, Machine Translation, Opinion Mining, etc., due to the irregularities found in such content. Secondly, non-native speakers of English, older Internet users and non-members of the in-groups often find such texts difficult to comprehend. Prompt use of Internet and the resulting noisy user generated text found in different social media platforms such as social networking sites, blogs, etc., cause a hindrance in understanding casual written English, which often does not conform to the rules of spelling, grammar and punctuation.

In this paper, we present an approach for text normalization of social media text. Our approach uses a sequence to sequence model (Sutskever et al., 2014) in which we tried Recurrent Neural Network (RNN) based encoder-decoder approach (Bahdanau et al., 2014) with Long Short Term Memory (LSTM). The use of LSTMs for text normalization in the present work is motivated by (Sproat and Jaitly, 2016). We take a character based LSTM approach motivated by the work of (Ling et al., 2015) who showed that character based approach is superior to word based approach for neural network based sequence to sequence modelling tasks[2]. Our LSTM model was trained with attention mechanism (Bahdanau et al., 2014).

2 Related Work

Text Normalization is a well known task in the field of NLP, particularly in the Social Media domain. Clark and Araki (2011) provides a detailed survey on the challenges and applications of text normalization in Social Media.

Researchers have shown that text normalization is a major factor in improving performance of NLP intermediate tasks like part-of-speech tagging (Han et al., 2013) and NLP applications like machine translation (Hassan and Menezes, 2013).

Research in text normalization started with spelling correction with noisy channel model (Kernighan et al., 1990; Mays et al., 1991). Since then several different approaches have been proposed by researchers. We report here a few of the most prominent works on text normalization, with a particular focus on social media.

[1] https://noisy-text.github.io/2015/norm-shared-task.html

[2] (Ling et al., 2015) showed improvement on the machine translation task. Text normalization is conceptually very similar to and can be modelled as a machine translation task using noisy–clean parallel corpus..

S Bandyopadhyay, D S Sharma and R Sangal. Proc. of the 14th Intl. Conference on Natural Language Processing, pages 312–321, Kolkata, India. December 2017.

Statistical Approaches: In the early 1990's, researchers in the AT&T Bell Labs (Kernighan et al., 1990) and IBM Research (Mays et al., 1991) carried out independent work on spelling correction using noisy channel model. This generative model has remained the most dominant and successful approach to text normalization until very recently.

Hassan and Menezes (2013) proposed an approach for normalizing social media text which used random walk framework on a contextual similarity bipartite graph constructed from n-grams sequences, which they interpolated with edit distance . They used the proposed method as a preprocessing step to improve machine translation quality on social media text.

Pennell and Liu (2010) proposed text normalization for text messages (SMS) to make them suitable as input to speech synthesizer. They used statistical classifier which tries to learn when and which character to delete and then reverse the mappings to normalize short text messages.

Sproat et al. (2001) proposed a taxonomy of non-standard words (NSW) and explored several methods like n-gram language models, decision trees, weighted finite state transducers, etc., for text normalization of NSWs. They reported that a systematic class-specific treatment results in improved text normalization.

Choudhury et al. (2007) proposed a Hidden Markov model based text normalization approach for SMS texts and texting language. Aw et al. (2006) used SMT models to normalize noisy SMS text by translating SMS text to Regular English text. Mikheev (2000) solved three major problems in text normalization: sentence boundary disambiguation, disambiguation of capitalized words when they are used in positions where capitalization is expected, and identification of abbreviations.

Deep Learning Based Approaches: Deep learning based approaches have emerged as a competitive alternative in recent years and research have been reported on deep learning based text normalization.

One of the most prominent work on deep learning based approach to text normalization is (Sproat and Jaitly, 2016) which proposed different RNN architectures to normalize texts for text-to-speech (TTS) Systems. The main focus of their work was to normalize *written texts* to their *correct spoken form*. The proposed system used LSTMs and attention based Sequence-To-Sequence models (Bahdanau et al., 2014). Sproat and Jaitly (2017) used the same framework to build their TTS text normalization models for English and Russian and trained their models on huge amount of training data. Their dataset consists of 1.1 billion English words and 290 million Russian words and they reported very high accuracy, over 0.99 for both English and Russian. They also augmented their system with a finite-state transducer (FST) filter to take care of mistakes made by the RNN based model.

Deep Neural Network models suffer from the *Out-Of-Vocabulary* (OOV) problem (Luong et al., 2014), when text normalization is performed using word based approach. Xie et al. (2016) solved this problem by illustrating how character based neural networks are much better in normalizing noisy and user generated texts. They also showed that results can be improved by introducing synthesized errors in a datasets. They showed improvement using noisy text collected from English learner forum.

Leeman-Munk et al. (2015) proposed a model for normalizing noisy text which uses two augmented feed forward networks (Glorot and Bengio, 2010), flagger to identify the word to be normalized and at last a normalizer which provides the correct output for one token at a time.

Chollampatt et al. (2016) showed that neural machine translation models are better in correcting grammatical errors, a task closely related to text normalization, as compared to phrase based statistical machine translation models (Wang et al., 2014).

Other noticeable works in text normalization include the use of adaptive parser-centric strategy (Zhang et al., 2013) to convert noisy texts into grammatically correct texts, unsupervised model using semantic similarity and Re-ranking strategy (Li and Liu, 2014). Torunoğlu and Eryiğit (2014) proposed a cascaded approach for normalizing Turkish text by dividing the main problem into sub problems and solving them one by one. Liu (2012) used character-block level SMT to normalize SMS and Twitter text. Pusateri et al. (2017) reports the use of bi-directional LSTM for the task of *inverse* text normalization, the objective of which is the exact opposite of (Sproat and Jaitly, 2016), i.e. to convert the spoken form token sequence produced by a speech recognizer into written form.

3 System Architecture

3.1 Recurrent Neural Network

The main motive behind using RNN for text normalization is to utilize sequential content. Input to the RNNs is the current information they see as well as the previous information remembered by them at that time. In Fig 1, taken from (Elman, 1991), "BTSXVPE" at the bottom shows the current input and "CONTEXT UNITS" represents the output at the previous step. The decision of recurrent net reached at time step t_i affects the decision it will reach one moment later at time step t_{i+1}. As discussed in (Quast, 2016), RNNs are called recurrent because they perform the same task for every element of a sequence, with the output being dependent on the previous computations. It is often said that recurrent networks have memory. Adding memory to neural networks has a purpose: there is information in the sequence itself and recurrent nets use it to perform tasks. That sequential information is preserved in the recurrent network's hidden state which manages to span many time steps as it cascades forward to affect the processing of each new example. This feature of RNN makes it an efficient approach for normalizing noisy text.

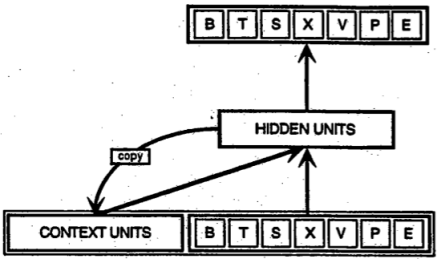

Figure 1: RNN Network Architecture

A drawback of the RNN is that, as the gap between the relevant information and the point where it is needed widens, RNNs fail in remembering the information. This problem with RNN was explored by Bengio et al. (1994). To overcome this problem, LSTM Networks were introduced by Hochreiter and Schmidhuber (1997).

3.2 Long-Short Term Memory Network

As discussed in (Olah, 2015), LSTM networks are another variety of RNNs, mainly used for learning long-term dependencies. In standard RNNs, there is always a chain of repeating modules containing a simple structure.

LSTMs are a fairly simple extension of RNNs. The objective of LSTM can be briefly described into three points as follows.

- **Deciding which information to remember/forget**: The model needs to learn a separate method to forget/remember information when new inputs come in; it needs to know which beliefs to keep and which ones to throw away.
- **Updating new information**: When new input comes in, the model first forgets any long-term information it decides it no longer needs. Then it learns which parts of the new input are worth using and saves them into its long-term memory.
- **Handling long-term dependencies**: Finally, the model needs to learn which parts of its long-term memory are immediately useful and therefore it needs to focus on.

3.3 RNN Encoder–Decoder

RNN Encoder–Decoder models (Cho et al., 2014) can be stated a sequence to sequence mapping between two sequences, learned using two RNNs, one on either side, encoder and decoder, which are trained jointly. For example, in our model, sentences having erroneous words were kept at the encoder side and the corresponding sentences having correctly spelled words were kept at the decoder side. The encoder part receives a sequence and then coverts it into an encoded representation of the sequence, which is further decoded by the decoder to provide the output sequence. RNN encoder-decoder models may contain different cells like GRU or LSTM, or simply RNN. It is quite obvious that the encoded representation of a sequence must be a fixed size vector.

The encoder encodes an input sequence into a fixed-length vector representation and the decoder decodes a given fixed-length vector representation into an output sequence.

3.4 Attention-based Bidirectional RNN Model

Attention Mechanisms in Neural Networks are (very) loosely based on the visual attention mechanism found in humans. Our bidirectional RNN encoder consists of forward and backward RNNs. Using attention mechanism, the decoder gives attention to different parts of the input sequence at each step of the output generation .

4 Data Sets and Resources

As discussed in Section 3.4, we trained a monolingual encoder-decoder based S2S model for our task, where the encoder encodes the input incorrect text and the decoder produces the correct text as output. We consider noisy texts and their corresponding corrected version as a parallel data on which we train our S2S model. However, it is a well known fact that deep neural network models (S2S in our case) typically require large amount of training data. Obtaining such large parallel training data, particularly for the social media domain is a major challenge itself in text normalization research. Therefore, in the absence of such parallel training data, we constructed synthetic data, i.e. artificially developed parallel dataset in which one side consists of sentences having misspelled and noisy words and the other side consists of sentences containing correctly spelled and normalized words. The noisy side (having misspelled words) of the parallel data was created by randomly replacing words in a clean text corpus with the help of four different dictionaries mentioned in Section 4.1. All of these dictionaries consist of parallel list (a hash table) of correct words and their corresponding incorrect noisy versions, where the correct word is stored as a *key* and the values for those keys are one or many misspelled words corresponding to the key.

4.1 Dictionaries

We used Peter Norvig Copus[3], one of the most popular resource in text normalization research, there were containing 7,841 correctly spelled words and their corresponding misspelled version(s). A snapshot of the Peter Norvig Corpus is shown below.

.....
looking: loking, begining, luing, look*2, locking, lucking, louk, looing, lookin, liking
eligible: eligble, elegable, eligable
scold: schold, skold
.....

However, the Peter Norvig corpus is a general domain spelling error corpus, i.e., it is not specifically designed for noisy social media content and as such does not contain spelling error phenomena that are typical to social media text. To include the flavour of errors which occur in social media conversations, we used two spelling error dictionaries provided by the WNUT 2015 shared task[4] on "Normalization of Noisy Text" which contain one to one mapping of incorrect words to correct words. The two dictionaries contain 3,804 and 41,182 [correct, incorrect] word pairs respectively. To make the dataset even noisier, we also constructed another social media spelling error dictionary containing 652 new word pairs by manually observing different Whatsapp Group chat conversations, public comments on Facebook's posts and Facebook Conversations.

4.2 Synthetic Dataset Preparation

We crawled 500K sentences from different news domain websites such as Fox News[5], The Guardian[6], Yahoo News[7] and CNN News[8] and then with the help of Peter Norvig's corpus we prepared a synthetic noisy dataset by replacing some words in each sentence with their corresponding misspelled words, if found in the Peter Norvig corpus. If there exist multiple misspelled versions for the same word in the Peter Norvig corpus, then the choice of misspelled word was taken randomly. Since many of the sentences were very long in the crawled corpus, we broke them (both the original sentence and the corresponding noisy sentence) down to sequences of five-grams in order to keep the sequence length shorter. Here sequence length refers to the number of characters in a sequence. Thus, we created a synthetic parallel dataset, $Synthetic_1$, and Table 1 shows how our parallel synthetic dataset look like, in which the misspelled words are shown as underlined.

We also created another synthetic dataset ($Synthetic_2$) using a Chat Conversation dataset[9] in a similar manner, however, with two noticeable changes. For this dataset we did not split the sequence into n-grams because in this dataset sequence length was not too large as compared to $Synthetic_1$ dataset and we took the help of all the four dictionaries to create the parallel dataset. The chat conversation dataset belongs to the Cornell Movie Dialogue dataset (Danescu-Niculescu-Mizil and Lee, 2011) and it contains conversational data extracted from movie scripts. The dataset was

[3] http://norvig.com/ngrams/spell-errors.txt

[4] https://noisy-text.github.io/2015/norm-shared-task.html

[5] http://www.foxnews.com/

[6] https://www.theguardian.com/international

[7] http://noornotews.yahoo.com/

[8] http://edition.cnn.com/

[9] https://github.com/1228337123/tensorflow-seq2seq-chatbot/tree/master/data

Raw Corpus	
... The government guidance will be reviewed early next year after a period of public comment ...	
Clean Text	**Noisy Text**
... The government guidance will be reviewed government guidance will be reviewed guidance will be reviewed early will be reviwed early next be reviewed early next year reviewed early next year after ...	... The govment guidence will be reviewed guverment guidence we'll be reviewed guidance wil be reviewed erly wiull be reviewed eigly next be reviewed erly enxt yeer reviewed erly nexst year afert ...

Table 1: A snapshot of the $Synthetic_1$ dataset

originally built for building chat systems. We constructed our synthetic data from this raw data. With $Synthetic_2$, our main motive was to build a very noisy dataset containing errors that are typical to social media. Therefore, we replaced as many words as we could find in the two WNUT dictionaries and our social media dictionary. At last, we checked if any of the non-replaced words in a sequence occurs in keys of the Peter Norvig corpus, and if found, they were also replaced with a corresponding (randomly chosen) misspelled word. Thus, we created the parallel dataset $Synthetic_2$ reflecting errors typical of social media text. A snapshot of the $Synthetic_2$ dataset is shown in Table 2.

Other than these synthetic datasets, we also used the standard training dataset released by the WNUT 2015 shared task on "Normalization of Noisy Text". This dataset consists of real Twitter data containing different types of abbreviations used and errors made in social media conversations.

Thus, we ended up with three different datasets[10] – $Synthetic_1$, $Synthetic_2$ and WNUT. Table 3 presents the dataset statistics of all the datasets that we used to train and evaluate our models.

5 Proposed Models

Model 1 (M_1) : This model was trained and tested on the $Synthetic_1$ training and test set, respectively. A batch size of 305 was considered and model was trained for 20 epochs at constant learning rate of 0.001. In this dataset sequence length was 74. Training this model took 20 hours (1 hour/epoch) on a single GPU. This model was built as a general purpose spelling corrector for regular English sentences.

Model 2 (M_2) : From this model onwards, all our models are focused mainly on in social media text. This model was trained and tested on the $Synthetic_2$ training and test set, respectively. The sequence length of was 214 and batch size was kept to 32. The model was trained for 4 epochs as after 4th epoch, loss function started diverging and after making changes in the learning rate, there was no sign of convergence of loss. Learning rate was kept constant for 4 epochs at 0.001. Training this model took 16 hours (4 hour/epoch) on a single GPU.

Model 3 (M_3) : This model was also trained and tested on $Synthetic_2$ dataset, however, the sequence length was kept to 160. We decreased the sequence length so that we could train our model for more number of epochs. Decreasing the sequence length made the training dataset smaller to 1,29,590 sentences, we refer to this dataset as $Synthetic_3$. We were able to train this model for 7 epochs by changing the learning rate form 0.001 to 0.0001 after 4 epochs. Training this model took 17.5 hours (2.5 hour/epoch) on a single GPU.

Model 4 (M_4) : This model was trained and tested on WNUT datasets. Sequence length was kept at 160 and model was trained for 50 epochs and batch size for this experiment was 32. Learning rate was changed from 0.001 to 0.0001 and then back to 0.001, in between the epochs after observing the behaviour of the loss function. Training this model took 2.5 hours (3 minutes/epoch) on a single GPU.

Model 5 (M_5): This model was trained on the merged training datasets of $Synthetic_3$ and

[10]We will release the synthetics datasets and make available for text normalization research upon publication of the paper.

Clean Text	Noisy Text
...	...
Not the hacking and gagging and spitting part.	Not tne hackinq und gaggin nd spittin part.
Please.', "You're asking me out.	Plz.', "You're askin meh out.
That's so cute. What's your name again?"	That's sou cute. What's yur nyam again?"
...	...

Table 2: A snapshot of $Synthetic_2$ dataset

Statistics								
Datasets	Training Set				Test Set			
	Sentences	Words			Sentences	Words		
		Incorrect	Correct	Total		Incorrect	Correct	Total
Synthetic_1	495,000	1,345,500	1,129,500	2,475,000	5,000	15,520	9,480	25,000
Synthetic_2	139,683	953,881	385,792	1,339,673	1,000	7,569	4,101	11,670
Synthetic_3	129,690	856,256	325,456	1,181,781	1,000	6,528	3,594	10,122
WNUT	2,950	19,903	27,482	44,385	1967	11,239	18,182	29,421

Table 3: Dataset Statistics

WNUT. The model was trained for 15 epochs. Loss function was constant at 0.001 and batch size was 256. Training this model took 18.75 hours (1.25 hours/epoch).

Model 6 (M_6): This model was built using transfer learning approach by using the trained weights of model M_3 for further training of the model on the WNUT dataset. Sequence length was kept at 160 and the model was trained for 10 epochs. The batch size was 32. The objective behind using transfer learning approach was to make use of the additional synthetic training dataset and hopefully to improve the system performance on the WNUT test set.

Model 7 (M_7): This is another model built using transfer learning approach, however, in this case training was first carried out on the WNUT datasets for 20 epochs and then the learned weights were further used to retrain the model on the $Synthetic_3$ dataset (10 epochs). Loss function for this model was constant at 0.001 and the batch size was 128. Training this model on the WNUT dataset and the $Synthetic_3$ dataset took 65 minutes (3.25 minutes/epoch) and 6.5 hours (40 minutes/epoch), respectively.

"Negative log Likelihood" was used as the loss function for all the experiments (M_1–M_7) as according to (Lewis and Gale, 1994), this loss function proved to be quite perfect for Sequence to Sequence models. "ADAM" optimizer as described in (Kingma and Ba, 2014) was used for all the experiments. All models mentioned here, used 3 layers on each side encoder & decoder.

6 Experimental Setup

Before loading the datasets for experiments, we padded the data. For faster computation, input sequences were divided into number of batches. Since there were variations in sequence length, we padded the data to make them all having uniform length. "EOS" token was kept at the end of each sequence, to identify its end. Shorter length sequences were padded with trailing zeros. We used word2index dictionary containing *key* as all characters and numbers as their indices. "EOS" and "PAD" were given "0" and "1" index respectively. The length of this dictionary was 98. The synthetic data and the dictionaries mentioned in Section 4.1 will be made publicly available for research upon publication of the paper.

Our models were built using deep learning library *Tensorflow*[11]. TensorFlow allows to efficiently perform specific machine learning number-crunching operations like derivatives on huge matrices . With Tensorflow, processing can be easily distributed across CPU cores, GPU cores, or multiple devices like multiple GPUs and even across a distributed network of computers. *Python*[12] was used for preparation of the dataset as mentioned in section 4.2 and scripting of the models.

7 Results

Since the proposed model can also mistakenly modify some correct words that should not be changed, precision and recall are the most suitable metrics

[11]https://www.tensorflow.org/

[12]https://www.python.org/

for evaluating this scenario (Powers, 2011). Accordingly, we evaluated the proposed models using precision, recall and F1-score. In the context of text normalization, true positives refer to cases where incorrect words are replaced by the correct words, and true negatives represent correct words being left as they are. False positives concern cases when the model replaces a correct word by an incorrect word. False negatives pertain two cases when the model either does not provide any correction or provides a wrong correction for an incorrect word. Table 4 shows the evaluation results of all the experiments mentioned in Section 5. For the benefit of comparison, Table 5 groups these results into two subsets - the ones evaluated on the synthetic test sets and the others evaluated on the WNUT test set. It is to be noted here that the models trained only with the synthetic data (M_1, M_2 and M_3) are not meant to be evaluated with the WNUT test set, however, for the sake of comparison, we also evaluated these models on the the WNUT test set.

Among the experiments carried out only with synthetic datasets, the best result (F1 Score = 0.9205) was achieved with M_1. The other models trained on other synthetic datasets, M_2 and M_3, could not achieve similar results since M_2 and M_3 could not be trained much due to their large sequence length. However, it is to be noted that only the Peter Norvig spelling error corpus was used on news domain data to introduce noise and prepare the $Synthetic_1$ dataset, while all the four dictionaries were employed to introduce noise in conversational data to prepare the $Synthetic_2$ dataset. Therefore, $Synthetic_2$ dataset is much more reflective of social media data and hence more challenging. Among M_2 and M_3, since M_3 was trained on shorter sequences and was also trained for more epochs, it was able to produce better performance than M_2.

Then we evaluated our models on the WNUT dataset, the only standard dataset available for text normalization research. Since the training dataset size was very small, the model (M_4) produced relatively low performance (F1 Score = 0.8223) even after training for 50 epochs. This relatively low performance can be attributed to the very small amount of training data (only 2,950 sentences) in the WNUT dataset; deep learning based models are known to perform poorly than traditional machine learning based methods on small training data. However, this result is only next to the best result (F1 Score = 0.8421) achieved in the WNUT 2015 shared task (Baldwin et al., 2015) in the constrained category.

M_5, trained on a merged training set of $Synthetic_3$ and WNUT, produced better results on both the $Synthetic_3$ as well as WNUT test sets, which can be observed by comparing the performance of M_5 with M_3 and M_4.

Both M_6 and M_7 make use of transfer learning approach. M_6 improves the model performance on the $Synthetic_3$ test set over M_3, however, it could not improve over M_5. On the other hand, M_6 could not beat the performance of M_4 on the WNUT test set. The reason behind this could be that M_6 is essentially a pre-trained model M_3, trained on $Synthetic_3$ and further trained on the WNUT dataset. It is to be noticed however that it provides huge improvement over M_3's performance on the WNUT testset.

We changed the sequence of training in M_7, i.e., first on WNUT and then on $Synthetic_3$ dataset, and also increased the batch size to 128. These changes improved the model performance significantly and provided the best performance on both the $Synthetic_3$ and WNUT test sets. It provided an F1 Score of 0.9098 on the WNUT test set which outperforms the best result (F1 Score = 0.8421) reported in the WNUT shared task (Baldwin et al., 2015).

Table 6 shows a comparison of the results obtained by our systems against the two top performing systems in the WNUT shared task in both constrained and unconstrained track. Surprisingly, unconstrained systems were not able to outperform constrained systems in the WNUT shared task, as is also noted in (Baldwin et al., 2015). However, by making use of our synthetic training data and using a transfer learning approach, we were able to obtain state of the art results on the WNUT dataset.

Our models were able to correctly normalize social media specific errors and abbreviations. For example *"LOL"* was normalized to *"Laughing out Loud"* or *"Lots of Laughs"* depending upon the context, *"GM"* was normalized to *"Good Morning"*, *"k"* to *"ok"* and so on. Among other types of social media errors, *"ohhhhhhhhhhhhhhh"* was normalized to *"oh"*, *"b4"* to *"before"*, *"hiiiiiiii"* to *"hi"*, etc.

Model	Train	Sequence Length	Test	Precision	Recall	F1 Score
M1	$Synthetic_1$	74	$Synthetic_1$	0.9622	0.8822	0.9205
			WNUT	0.1845	0.1756	0.1784
M2	$Synthetic_2$	214	$Synthetic_2$	0.8066	0.7204	0.7610
			WNUT	0.2569	0.2356	0.2457
M3	$Synthetic_3$	160	$Synthetic_3$	0.9102	0.8595	0.8841
			WNUT	0.3096	0.3156	0.3125
M4	WNUT	160	WNUT	0.8558	0.7915	0.8223
M5	Merged datasets ($Synthetic_3$ + WNUT)	160	$Synthetic_3$	0.9366	0.9089	0.9225
			WNUT	0.8569	0.8698	0.8633
M6	Transfer Learning $Synthetic_3 \rightarrow WNUT$	160	$Synthetic_3$	0.9389	0.8856	0.9114
			WNUT	0.8747	0.7260	0.7935
M7	Transfer Learning WNUT $\rightarrow Synthetic_3$	160	$Synthetic_3$	0.9458	0.9056	0.9252
			WNUT	0.9256	0.8945	0.9098

Table 4: Results of different models on synthetic and WNUT datasets

	Synthetic Test Sets		
	Precision	**Recall**	**F1 Score**
M1	0.9622	0.8822	0.9205
M2	0.8066	0.7204	0.7610
M3	0.9102	0.8595	0.8841
M5	0.9366	0.9089	0.9225
M6	0.9389	0.8856	0.9114
M7	0.9458	0.9056	0.9252
	WNUT Test Set		
	Precision	**Recall**	**F1 Score**
M4	0.8558	0.7915	0.8223
M5	0.8569	0.8698	0.8633
M6	0.8747	0.7260	0.7935
M7	0.9256	0.8945	0.9098

Table 5: Results of RNN based LSTM Models on Synthetic and WNUT Test Sets

8 Conclusions & Future work

In this paper we presented a work on text normalization using RNN based encoder-decoder LSTM with an attention mechanism. We illustrate the usefulness of our approach on a variety of noisy datasets - standard real dataset as well as synthetic datasets. We obtained state of the art results on the WNUT dataset using our synthetic training data and a transfer learning approach. Another important contribution of this study is the development of noisy–clean parallel synthetic datasets from user-generated text reflecting both error patterns in regular text as well as social media. The proposed model, with further improvisations, can be useful in the field of social media to make social media communications better and more understandable.

Our next goal is to construct a large real (i.e., not synthetic) social media dataset, similar to WNUT, suitable for training deep learning models which will definitely help to improve text normalization of social media texts. We would also like to explore other deep learning based models for the task.

Acknowledgments

We would like to thank the anonymous reviewers for their feedback. Sudip Kumar Naskar is supported by Media Lab Asia, MeitY, Government of India, under the Young Faculty Research Fellowship of the Visvesvaraya PhD Scheme for Electronics & IT.

References

Aw, A., Zhang, M., Xiao, J., and Su, J. (2006). A phrase-based statistical model for sms text normalization. In *Proceedings of the COLING/ACL on Main conference poster sessions*, pages 33–40. Association for Computational Linguistics.

Bahdanau, D., Cho, K., and Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*.

Baldwin, T., De Marneffe, M. C., Han, B., Kim, Y.-B., Ritter, A., and Xu, W. (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*

Mode	System	Precison	Recall	F1 Score
Constrained	**NCSU_SAS_NING**	0.9061	0.7865	**0.8421**
	NCSU_SAS_WOOKHEE	0.9136	0.7398	0.8175
	M_4	0.8558	0.7915	0.8223
Unconstrained	**IHS_RD**	0.8469	0.8083	**0.8272**
	USZEGED	0.8606	0.7564	0.8052
	M_5	0.8569	0.8698	*0.8633*
	M_6	0.8747	0.7260	0.7935
	M_7	0.9256	0.8945	*0.9098*

Table 6: Comparison of results between our systems and top two systems of the WNUT shared task

Noisy User-generated Text (WNUT 2015), Beijing, China.

Bengio, Y., Simard, P., and Frasconi, P. (1994). Learning long-term dependencies with gradient descent is difficult. *IEEE transactions on neural networks*, 5(2):157–166.

Cho, K., Van Merriënboer, B., Gulcehre, C., Bahdanau, D., Bougares, F., Schwenk, H., and Bengio, Y. (2014). Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078*.

Chollampatt, S., Taghipour, K., and Ng, H. T. (2016). Neural network translation models for grammatical error correction. *arXiv preprint arXiv:1606.00189*.

Choudhury, M., Saraf, R., Jain, V., Mukherjee, A., Sarkar, S., and Basu, A. (2007). Investigation and modeling of the structure of texting language. *International journal on document analysis and recognition*, 10(3):157–174.

Clark, E. and Araki, K. (2011). Text normalization in social media: progress, problems and applications for a pre-processing system of casual english. *Procedia-Social and Behavioral Sciences*, 27:2–11.

Danescu-Niculescu-Mizil, C. and Lee, L. (2011). Chameleons in imagined conversations: A new approach to understanding coordination of linguistic style in dialogs. In *Proceedings of the 2Nd Workshop on Cognitive Modeling and Computational Linguistics*, CMCL '11, pages 76–87, Stroudsburg, PA, USA. Association for Computational Linguistics.

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

Glorot, X. and Bengio, Y. (2010). Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the Thirteenth International Conference on Artificial Intelligence and Statistics*, pages 249–256.

Han, B., Cook, P., and Baldwin, T. (2013). Lexical normalization for social media text. *ACM Transactions on Intelligent Systems and Technology (TIST)*, 4(1):5.

Hassan, H. and Menezes, A. (2013). Social text normalization using contextual graph random walks. In *ACL (1)*, pages 1577–1586.

Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural computation*, 9(8):1735–1780.

Kernighan, M. D., Church, K. W., and Gale, W. A. (1990). A spelling correction program based on a noisy channel model. In *Proceedings of the 13th Conference on Computational Linguistics - Volume 2*, COLING '90, pages 205–210, Stroudsburg, PA, USA. Association for Computational Linguistics.

Kingma, D. and Ba, J. (2014). Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*.

Leeman-Munk, S., Lester, J., and Cox, J. (2015). Ncsu_sas_sam: Deep encoding and reconstruction for normalization of noisy text. In *Proceedings of the Workshop on Noisy User-Generated Text at ACL, Beijing, China*, pages 154–61.

Lewis, D. D. and Gale, W. A. (1994). A sequential algorithm for training text classifiers. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 3–12. Springer-Verlag New York, Inc.

Li, C. and Liu, Y. (2014). Improving text normaliza-

tion via unsupervised model and discriminative reranking. In *ACL (Student Research Workshop)*, pages 86–93.

Ling, W., Trancoso, I., Dyer, C., and Black, A. W. (2015). Character-based neural machine translation. *arXiv preprint arXiv:1511.04586*.

Liu, Y. (2012). Improving text normalization using character-blocks based models and system combination.

Luong, M.-T., Sutskever, I., Le, Q. V., Vinyals, O., and Zaremba, W. (2014). Addressing the rare word problem in neural machine translation. *arXiv preprint arXiv:1410.8206*.

Mays, E., Damerau, F. J., and Mercer, R. L. (1991). Context based spelling correction. *Inf. Process. Manage.*, 27(5):517–522.

Mikheev, A. (2000). Document centered approach to text normalization. In *Proceedings of the 23rd annual international ACM SIGIR conference on Research and development in information retrieval*, pages 136–143. ACM.

Olah, C. (2015). Understanding lstm networks.

Pennell, D. L. and Liu, Y. (2010). Normalization of text messages for text-to-speech. In *Acoustics Speech and Signal Processing (ICASSP), 2010 IEEE International Conference on*, pages 4842–4845. IEEE.

Powers, D. M. (2011). Evaluation: from precision, recall and f-measure to roc, informedness, markedness and correlation.

Pusateri, E., Ambati, B. R., Brooks, E., Platek, O., McAllaster, D., and Nagesha, V. (2017). A mostly data-driven approach to inverse text normalization. *Proc. Interspeech 2017*, pages 2784–2788.

Quast, B. (2016). rnn: a recurrent neural network in r. *Working Papers*.

Sproat, R., Black, A. W., Chen, S., Kumar, S., Ostendorf, M., and Richards, C. (2001). Normalization of non-standard words. *Computer speech & language*, 15(3):287–333.

Sproat, R. and Jaitly, N. (2016). Rnn approaches to text normalization: A challenge. *arXiv preprint arXiv:1611.00068*.

Sproat, R. and Jaitly, N. (2017). An rnn model of text normalization. *Proc. Interspeech 2017*, pages 754–758.

Sutskever, I., Vinyals, O., and Le, Q. V. (2014). Sequence to sequence learning with neural networks. In *Advances in neural information processing systems*, pages 3104–3112.

Torunoğlu, D. and Eryiğit, G. (2014). A cascaded approach for social media text normalization of turkish. In *Proceedings of the 5th Workshop on Language Analysis for Social Media (LASM)*, pages 62–70.

Wang, Y., Wang, L., Zeng, X., Wong, D. F., Chao, L. S., and Lu, Y. (2014). Factored statistical machine translation for grammatical error correction. In *CoNLL Shared Task*, pages 83–90.

Xie, Z., Avati, A., Arivazhagan, N., Jurafsky, D., and Ng, A. Y. (2016). Neural language correction with character-based attention. *arXiv preprint arXiv:1603.09727*.

Zhang, C., Baldwin, T., Ho, H., Kimelfeld, B., and Li, Y. (2013). Adaptive parser-centric text normalization. In *ACL (1)*, pages 1159–1168.