Attention-based Sequence to Sequence Model for Tweet Normalization

AAAI Press
Association for the Advancement of Artificial Intelligence
2275 East Bayshore Road, Suite 160
Palo Alto, California 94303

Abstract

Paraphrase identification for tweet is a new challenge due to informal expression. In this paper, we present a end-to-end approach to sequence learning that transfers text normalization into a neural machine translation task. Then a siamese recurrent architecture is used for learning sentence similarity after the sentences have been normalized. Our model employs a convolutional neural network and a highway network over characters, whose output is given to a long short-term memory recurrent neural network to map the input sequence to a vector of a fixed dimensionality, and then another long short-term memory to decode the target sequence from the vector. We add an attentional mechanism to improve the performance of tweet normalization, which can be beneficial to paraphrase detection indirectly. Novelty of this work is three-fold: First, to the best of our knowledge, this is an early attempt to adopt attention-based end-to-end model for tweet normalization. Second, our proposed model relies on character-level inputs, making it fewer parameters to obtain comparable performance to word inputs. And third, we conduct a series of experiments and prove that the proposed method is advantageous over the state-of-art solutions for tweet normalization and semantic similarity. Our method shows significant performance gains on the paraphrase identification in tweet task compared with several strong baselines.

Introduction

Semantic similarity of text are used in many semantic tasks such as paraphrase identification (Xu et al. 2014), textual entailment (Henderson and Popa 2016), and question answering (Severyn and Moschitti 2015). Twitter engages millions of users, who naturally talk about the same topics simultaneously and frequently convey similar meaning using diverse linguistic expressions. The unique characteristics of this user-generated text presents new challenges and opportunities for semantic similarity measurement. Unfortunately, traditional natural language processing methods sometimes perform poorly when processing this kind of text. One reason is that tweets are very informal, and contain many misspelled words, abbreviations and many other non-standard tokens, which make it substantially different from formal written text. To improve the performance on the social media semantic similarity, it is inevitable to leverage normalization techniques which can automatically convert the non-standard tokens into the corresponding standard words.

Intuitively, tweet normalization will be beneficial for paraphrase identification and textual semantic similarity. For example, if 'tmr' is converted to 'tomorrow', it will improve the semantic representation for tweets and promote similarity comparison performance between text pairs. This normalization task has received an increasing in social media language processing. However, most of previous work on normalization assumed that they already knew which tokens are non-standard words (NSW) that need normalization. Then different methods are applied only to these tokens. Han and Baldwin (2011) is the typical work which made a pilot reasearch on NSW identification. One straight forward method to do this is to use a dictionary to classify a token into in-vocabulary (IV) words and out-of-vocabulary (OOV) words, and just treat all the OOV words as NSW. Contrast to straight forward work, joint part-of-speech (POS) and text normalization was proposed by Li and Liu (2015), the objective of this work is to perform POS tagging and text normalization at the same time. It indicates the thought that joint tweet normalization and similarity.

Character-aware neural language model (Kim et al. 2016) relies only on character-level inputs and predictions are still made at the word-level, and the model can leverage subword information through a character-level convolutional neural network (CNN) and have a good performance on the estimation of rare words embeddings. Therefore, it is attracting to explore neual language model for tweet normalization. Xu et al. (2015) proposed a share task evaluation on both paraphrase identification and semantic textual similarity for Twitter data. Although some works (Zarrella et al. 2015; van der Goot and van Noord 2015) take tweet normalization into account, this researches are limited to replace the OOV words with normalization dictionary.

In this paper, we proposed a character-level neural language model for tweet normalization, and then measure the semantic of normalized text for paraphrase identification and textual similarity. Our work investigates two methods using normalized tweets for sematic similarity measurement. One adopts a pipline strategy, and the other uses a joint fashion. Our experimental results demonstrate that our proposed model gives a significant performance improvement on NSW detection compared with the dictionary baseline

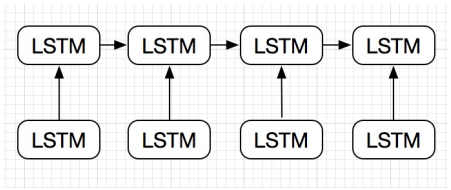

Figure 1: The structure of model

system and our proposed joint model performs better than the pipeline method, and it outperforms the state-of-the-art paraphrase identification system. To summarize, our contributions in this paper are as follows:

- We proposed a character-level neural language model for tweet normalization, which achieve results on par with the existing state-of-the-art NSW detection.
- The proposed joint model can combine the normalization and semantic textual similarity techniques to improve the fprmance of these two tasks on English social media data.
- We demonstrate the effectiveness of our proposed method. Experimental results achieve the state-of-the-art performance on normalization and similarity.

Model

The proposed model adopt a sequence to sequence method for learning a neural language model. We illustrate these two model types in Figure 1.

The model includes two parts, one is normalization, and other is similarity.

The question that inspired this paper was whether neural language model could also benefit for text normalization.; that is ... To answer this question we introduce character-aware neural language model for tweets normalization. **Draw the figure of model.**

Long Short-Term Memory

Recurrent neural networks (RNNs) are able to process input sequences of arbitrary length via the recursive application of a transition function on a hidden state vector h_t. At each time step *t*, the hidden state h_t is a function of the input vector x_t that the network receives at time *t* and its previous hidden state h_{t-1}. The hidden state $h_t \in \mathbf{R^d}$ can be interpreted as a *d*-dimensional distributed representation of the sequence of tokens observed up to time *t*. Commonly, the RNN transition function is an affine transformation followed by a pointwise non-linearity such as the hyperbolic tangent function:

$$h_t = \tanh(Wx_t + Uh_{t-1} + b) \tag{1}$$

However, a problem with RNNs with transition functions of this form is that during traning , components of the gradient vector can grow or decay exponentially over long sequences (Hochreiter 1998). This problem with exploding or vanishing gradients makes it difficult for the RNN model to learn long-distance correlations in a sequence.

The LSTM architecture addresses this problem of learning long-term dependencies by introducing a memory cell that is able to preserve state over long periods of time (Hochreiter and Schmidhuber 1997). While numerous LSTM variants have been described, here we describe the version used by Zaremba and Sutskever (2014). We define the LSTM unit at each time step *t* to be a collection of vectors in R^d: an input gate i_t, a forget gate f_t, an output gate o_t, a memory cell c_t and a hidden state h_t. The entries of the gating vectors i_t, f_t and o_t are in [0, 1]. We refer to *d* as the memory dimension of the LSTM. The LSTM transition equations are the following:

$$i_t = \sigma(W^{(i)}x_t + U^{(i)}h_{t-1} + b^{(i)}) \tag{2}$$

$$f_t = \sigma(W^{(f)}x_t + U^{(f)}h_{t-1} + b^{(f)}) \tag{3}$$

$$o_t = \sigma(W^{(o)}x_t + U^{(o)}h_{t-1} + b^{(o)}) \tag{4}$$

$$u_t = \tanh(W^{(u)}x_t + U^{(u)}h_{t-1} + b^{(u)}) \tag{5}$$

$$c_t = i_t \odot u_t + f_t \odot c_{t-1} \tag{6}$$

$$h_t = o_t tanh(c_t) \tag{7}$$

where x_t is the input at the current time step, σ denotes the logistic sigmoid function and $\odot$ denotes elementwise multiplication. Intuitively, the forget gate controls the extent to which the previous memory cell is forgotten, the input gate controls how much each unit is updated, and the output gate controls the exposure of the internal memory state. The hidden state vector in an LSTM unit is therefore a gated, partial view of the state of the unit's internal memory cell. Since the value of the gating variables vary for each vector element, the model can learn to represent information over multiple time scales.

Neural Language Model

The goal of the LSTM is to estimate the conditional probability $p(y_1, ..., y_{T'}|x_1, ..., x_T)$ where $(x_1, ..., x_T)$ is an input sequence and $y_1, ..., y_{T'}$ is its corresponding output sequence whose length $T^{'}$ may differ from *T*. The LSTM computes this conditional probability by first obtaining the fixed-dimensional representation *v* of the input sequence $(x_1, ..., x_T)$ given by the last hidden state of the LSTM, and then computing the probability of $y_1, ..., y_{T'}$ with a standard LSTM language model formulation whose initial hidden state is set to the representation *v* of $x_1, ..., x_T$:

$$p(y_1, ..., y_{T'}|x_1, ..., x_T) = \prod_{t=1}^{T'} p(y_t|v, y_1, ..., y_{t-1}) \tag{8}$$

In this equation, each $p(y_t|v, y_1, ..., y_{t-1})$ distribution is represented with a softmax over all the words in the vocabulary. Note that we require that each sentence ends with a special end-of-sentence symbol "<EOS>", which enables the model define a distribution over sequences of all possible lengths.

Attention-Based Encoder

Attention-based contextual encoder that constructs a representation based on the generation context. Attention models adopt a look-back strategy by linking the current decoding stage with input sentences in an attempt to consider which part of the input is most responsible for the current decoding state.

Let *H*= $\{h_1^s(e), h_2^s(e), ..., h_N^s(e)\}$ be the collection of sentence-level hidden vectors for each sentence form the inputs, outputed from LSTM sentence encoder. Each element in *H* contains information about input sequences with a strong focus on the parts surrounding each specific sentence (time-step). During decoding, suppose that e_t^s denotes the sentence-level embedding at current step and that h_{t-1}^s(dec) denotes the hidden vector outputted from LSTM sentence decode at previous time step *t*-1. Attention models would first link the current step decoding information, i.e., h_{t-1}^s(dec) which is outputted form LSTM with each of the input sentence $i \in [1, N]$, characterized by a strength indicator v_i:

$$v_i = U^T f(W_1 \cdot h_{t-1}^s(dec) + W_2 \cdot h_i^s(enc)) \quad (9)$$

$W_1, W_2 \in \mathbb{R}^{K \times K}$, $U \in \mathbb{R}^{K \times 1}$. v_i is then normalized:

$$a_i = \frac{\exp(v_i)}{\sum_{i'} \exp(v_i^{'})} \quad (10)$$

The attention vector is then created by averaging weights over all input sentences:

$$m_t = \sum_{i \in [1, N_D]} a_i h_i^s(enc) \quad (11)$$

LSTM hidden vectors for current step is then achieved by combining c_t, e_t^s and h_{t-1}^s(dec):

$$c_t = f_t \cdot c_{t-1} + i_t \cdot l_t \quad (12)$$

$$h_t^s = o_t \cdot c_t \quad (13)$$

where $W \in \mathbb{R}^{4K \times 3K}$. h_t is then used for word predicting as in the vanilla version of the hierarchical model.

Training

The lack of generation constrants makes it posible to train the model on arbitrary input-output pairs. Once we have defined the local conditional model, $p(y_{i+1}|\text{x}, y_c; \theta)$, we can estimate the parameters to minimize the negative log-likelihood of a set of summaries. Define this training set as consisting of *N* input-output pairs ($\mathbf{x}^{(1)}$, $\mathbf{y}^{(1)}$),...,($\mathbf{x}^{(N)}$, $\mathbf{y}^{(N)}$). The negativ log-likelihood conveniently factors into a term for each token in the decode sentence:

$$NLL(\theta) = -\sum_{j=1}^{J} \log p(\mathbf{y}^{(j)}|\mathbf{x}^{(j)}; \theta) \quad (14)$$

We minimize NLL by using mini-batch stochastic gradient descent.

Attention

In parallel, the concept of "attention" has gained popularity recently in training neural networks, allowing models to learn alignments between different modalities. In the context of NMT, Bahdanaul et al. (2014) as successfully applied such attentional mechanism to jointly translate and align words. To the best of our knowledge, there has not been any other work exploring the use of attention-based architectures for NMT.

Word, Character, attention, sequence to sequence, language model, tweet normalization, semantic similarity, end-to-end.

Normalization

Given a piece of microtext, our model will normalize terms one bye one. Thus, the challenge is to determine the corresponding standard form *t*, which can be a term or a sequence of terms, for each observed term *t'*. Notice that *t* many or may not be the same with *t'*. *t'* is a NSW if $t' \neq t$. Thus, the task is to find the most probable normalization $t*$ for each observed term *t'*:

$$t* = argmax_t P(t|t') = argmax_t P(t'|t)P(t) \quad (15)$$

P(t'|t) models the noisy channel through which an intended term *t* is sent and is corrupted into the observed term *t'*. P(t) models the source that generates the intended term *t*. In practice, the source model can be approximated with an n-gram statistical language model.

Since a non-standard term and its corresponding standard form might be similar from one or multiple perspective(s), it is reasonable to assume that there are several channels, each of which would distort an intended term in one of the above aspects. More specifically, a grapheme channel would be responsible for the spelling distortion, a phoneme channel would cause phonetic corruptions, a context channel may shrink a sequence of terms into one term. In reality, an intended word may be transferred through one or multiple channels, and an observed term might be mixed corruption from more than one channel. Under this assumption, and letting $\{c_k|c_k \in C\}$ denotes the set of channels, so the aforementioned equation can be further developed to

$$t* = argmax_t \sum_k P(t', c_k|t)P(t) = argmax_t \sum_k P(t'|c_k, t)P(c_k|t)P($$

(16)

A term *t* is transfered through channel c_k with probability P(c_k|t). Within the channel c_k, term *t* is distorted to term *t'* according to the channel model P(t'|c_k, t).

Many approaches to text normalization adopt the noisy channel setting, where the model normalization source sentence *s* into target canonical form *t* is factored into two parts $\hat{t}$ = $\text{argmax}_t P(t)P(s|t)$. The error term $P(s|t)$ models hwo canonical strings are transformed into variants such as misspellings or abbreviations. The language model $P(t)$ encodes which target strings are probable. When large-scale normalized texts have been trained with language model, the weights of the model are obtained with the normalization corpus. When the informal texts are inputed into model,

the decoder of the proposed model will output the normalized text sequence and the informall words will be changed following the context information, which will be expressed with normalized words form normalized corpus dictionary. **AAAI2011Normalizing Microtext**

Similarity Measurement

We adopt a siamese architecture for learning sentence similarity. There are two networks LSTM_a and LSTM_b which each process one of the sentences in a given pair, but our networks architecture tied weights such that LSTM_a equals LSTM_b in this work. Nevertheless, the general united version of this model may be more useful for applications with asymmetric domains such as information retrieval. The LSTM learns a mapping from the space of variable length sequences of $\text{d}_i n$-dimensional vectors into $\mathbb{R}^{d_{rep}}$. More concretely, each sentence (represented as a sequence of word vectors) $x_1, \cdots, x_T$, is passed to the LSTM, which updates its hidden state at each sequence-index. The final representation of the sentence is encoded by $h_T \in \mathbb{R}^{d_{rep}}$, the last hidden state of the model. For a given pair of sentences, our approach applies a pre-defined similarity function g: $\mathbb{R}^{d_{rep}} \times \mathbb{R}^{d_{rep}} \rightarrow \mathbb{R}$ to their LSTM-representations. Similarites in the representation space are subsequently used to infer the sentences' underlying semantic similarity.

Note that unlike typical language model RNNs, which are used to predict the next word given the previous text, our LSTMs simply function like the encoder of sequence to sequence model. Thus, the sole error signal backpropagated during training stems from the similarity between sentence representations $h_{T_a}^{(a)}$, $h_{T_b}^{(b)}$, and how this predicted similarity deviate from the human annotated ground truth relatedness. We restrict ourselves to the simple similarity function: (*subscript must be changed as* $h_{T_1}^{(1)}$, or it is same as SiamStru(AAAI16).)

$$sim(h_{T_a}^{(a)}, h_{T_b}^{(b)}) = \exp(-||h_{T_a}^{(a)} - h_{T_b}^{(b)}||_1) \in [0, 1] \quad (17)$$

This forces the LSTM to entirely capture the semantic difference during training, rather than supplementing the RNN with a more complex learner that can help resolve shorcoming in the learned representations.

Experiments

We evaluated our model on the tweet normalization and paraphrase identification task, which is a benchmark for evaluating many semantic similarity models.

Dataset

The following data sets are used in our experiments. It includes tweet normalization and paraphrase identification.

- Data 1: 2,333 unique pairs of informal tokens and normalized words, collected form 2,577 Twitter messages, which selected form the Edinburgh Twitter corpus (Pennell and Liu 2011). And some changes were made on this data[1] by Li and Yang (2014). This data is used for training the normalization models for tweet normalization.
- Data 2: The data[2] was created by Chen and Liu (2015), which is used for joint POS tagging and normalization of non-standard tokens. The data contains two parts: one is from, which includes 549 tweets. The another contains 789 tweets. We removed the POS tagging and use the first as test data and the second as valid data.
- Data 3: We use the PIT corpus[3] for evaluating paraphrase identification. The training and development set contain 17,790 sentence pairs and test set includes 972 sentence pairs(Xu, Callison-Burch, and Dolan 2015). The data set can be used for paraphrase identification and semantic similarity in Twitter.

Traing Detail

Existing research has shown that attention mechanism work better than shallow recurrent neural network for squence to sequence tasks (Luong, Pham, and Manning 2015; Sutskever, Vinyals, and Le 2014). We adopt attention-based end-to-end structure with three layer for encoding and three layer for decoding, each of which is comprised of a different set of parameters. Each LSTM layers consists of 500 hidden neurons and the dimensionality of word embeddings is set to 500. Some other training details given as following.

- Input sentence sequences are reversed.
- Parameters are initialized form a uniform distribution between [-0.1, 0.1].
- Batch size is set to 16.
- Stochastic gradient descent is implemented without momentum using a fixed learning rate of 1. We start halving the learning rate every half epoch after 5 epochs. We trained our models for a total of 20 epoches.
- Dropout rate is set to 0.3.
- Size of the character embeddings is set to 30.
- Number of covolution filters is 500.

Our implementation on a single GPU[4] processes a speed of approximately 5,000-6,000 tokens per second.

Results

A summay of our experimental results for tweet normalization is given Table 1. We observe better performances for tweet normalization based on character-level input than the word-level with attention mechanism. In the Table, the experimental results include two parts input: word-level (Word) and character-level (Char). In each method, we use LSTM, attention (Attn) and reverse the source sequence (Rev), which is crucial to achieving good performance (Sutskever, Vinyals, and Le 2014). However, pre-trained word embeding is blind to subword information, and representation of rare words can be poorly estimated, leading to high perplexities for rare words. This is especially problematic in morphologically rich languages with long-tailed

[1] http://www.hlt.utdallas.edu/ chenli/normalization

[2] http://www.hlt.utdallas.edu/ chenli/normalization_pos/

[3] http://www.cis.upenn.edu/ xwe/ semeval2015pit/

[4] The GPUs with 1280 Cuda cores and 6GB memory.

Method	Accuracy
First-order Markov Viterbi	0.761
Secod-order Markov model	0.770
Joint Decoding	**0.773**
Word+LSTM	0.714
Word+LSTM+Rev	0.725
Word+LSTM+Attn	0.732
Word+LSTM+Rev+Attn	0.746
Char+CNN+LSTM	0.742
Char+CNN+HW+LSTM	0.751
Char+CNN+HW+LSTM+Rev	0.765
Char+CNN+HW+LSTM+Rev+Attn	0.771

Table 1: Tweet normalization results on test set.

frequency distribution of domains with dynamic vocabularies (e.g. Twitter). So we adopta character-level convolutional neural networks for leverages subword infomration, which output is used as an input to a long short-term memory network. Similar to the memory cells in LSTM networks, highway (HW) layers allow for training of deep networks by adaptively carrying some dimensions of the input directly to the output. In our method, a max-over-time pooling operation is applied to obtain a fixed-dimensional representation of the word, which is given to the highway network. The highway network' output is used as the input to LSTM. From the table 1, we observe that the proposed method can obtain the comparable performance to joint decoding for tweet normalization. In addition, the character-level input has significant performance on word-level embedding input. Meanwhile, the attentional mechanism is beneficial to sequence to sequence model.

After tweet normalization, the model has capability for normalizing the informal text. So we utilize the model with same parameters and weight for similarity evaluation. Both paraphrase identification and semantic similarity are used for evaluating the performance of tweet normalization. The paraphrase identification results are presented in Table 2. There are precesion(*Prec*), recall(*Rec*) and F-measure(F_1) for evaluating the performance of proposed model. There are four baselines, including logistic regression (Das and Smith 2009), weighted matrix factorization (Guo and Diab 2012), referential machine translation (Bicici 2015) and recursive auto-encoder (Socher et al. 2011). From the Table 2, we conclude that our method with character-level (Char) input is better than word-level (Word) input on paraphrase detection and the methods with normalization (Norm) have higher F_1 than the methods without normalization. What is more, our four methods obtain significant performance than other four baselines. The results indicates that normalization can promote the semantic representation of informal text.

We also implement semantic similarity experiment on PIT, the results are given in Table 3, the pearson is used for semantic similarity. As the same as the paraphrase identification, we adopt four method, including word-level and character-level input with optional tweet normalization, for evaluating the performance of the proposed methods.

Methods	*Prec*	*Rec*	F_1
Logistic Regeression	0.679	0.520	0.589
Recursive Auto-encoder	0.543	0.394	0.457
Weighted Matrix Factorization	0.450	0.663	0.536
Referential Machine Translation	0.859	0.417	0.562
Our Method (Word) without Norm	0.598	0.631	0.614
Our Method (Word) with Norm	0.642	0.651	0.646
Our Method (Char) without Norm	0.637	0.649	0.643
Our Method (Char) with Norm	0.654	0.672	**0.663**

Table 2: Test set results on PIT for paraphrase identification.

Model	*Pearson*
Logistic Regeression	0.511
Two-layer Neural Network	0.545
Weighted Matrix Factorization	0.350
Recurrent Neural Network	0.619
Our Method (Word) without Norm	0.581
Our Method (Word) with Norm	0.604
Our Method (Char) without Norm	0.613
Our Method (Char) with Norm	**0.628**

Table 3: Test set results on PIT for semantic similarity.

In addition, we also use four baselines, including logistic regeression (Das and Smith 2009), Two-layer Neural Network (Bertero and Fung 2015), Weighted Matrix Factorization (Guo and Diab 2012) and Recurrent Neural Network (Zarrella et al. 2015) for comparing the experimental performance. From the Table 3, we conclude that our model adopted the character-level input with normalization obtains the highest pearson score and word-level input also has better performance than other four baseline methods.

Related Work

Neural language models (NLM) contain a rich family of neural network architectures for language modeling. Some examples encompass recurrent (Mikolov et al. 2010), convolutional (Wang et al. 2015), and convolutional-recurrent (Kim et al. 2016). In Twitter, there are existing rare and unknown words due to the social media characteristic, therefore it is necessary for language model to point the unknown words before normalization. A neural network models using attention adopted two softmax layers in order to predict the next word in conditional language models (Gulcehre et al. 2016). Recurrent memory network (Tran, Bisazza, and Monz 2016) not only amplifies the power of recurrent neural networks but also facilitates the understanding of its internal functions, the model can learn the correct words by the trained language model. Log-bilinear language models integrate compositional morphological representations and probabilistic language modelling (Jan and Blunsom 2014).

Character-aware neural language model (Kim et al. 2016) can capture morphological features based on character-level input and the output is still word-level. Because the social media has long-tailed frequency distributions or do-

mains with dynamic vocabularies, so the model can leverage subword information for tweets normalization. The existing works for semantic similarity focus on sequence representation, and the model structure is usually a siamese architecture, which represent sentence pairs and compute the semantic similarity between pairs (Mueller and Thyagarajan 2016). Convolutional neural networks are used for representing sentence and learn multigranular sentence representations for paraphrase identification was proposed (Yin and Schütze 2015). The deep learning architecture based on convolutional networks can obtain mutliple levels of granularity, including unigram, short-ngram, long-ngram and sentence-level features, which are computed for similarity comparation for paraphrase detection.

Although normalization on informal text has potential function on sentence representation and semantic similarity, paraphrase can also improve the normalization performance (Ling et al. 2013) on parallel microblog data. Applying alignment feature for semantic similarity of text can capture both local ifnormation like lexical semantics and structural information like syntactic structures (Liang and Paritosh 2016). The method can avoid conventional fixed-length sentence representation and add more feature information for similarity measurement. Some works (Wieting et al. 2015; Pavlick et al. 2015a) consider the utilization of paraphrse database (Ganitkevitch, Van Durme, and Callison-Burch 2013; Pavlick et al. 2015b)for sentence similarity, entialment, and sentiment classification.

Our work apply character-level input style convolutional recurrent neural network model for trianing a language model on word-level prediction. After pre-trained with formal corpus such as Wiki or Penn Treebank data, the language model is used for normalizing the informal tweets, which can automatically point unknown or non-standard words and normalize them. The normalized tweets will be measured semantic similarity further, and the similarity will feedback for normalization based on assumptaion that paraphrasing sentence pairs will keep semantic equivalence. The experimental results demonstrate that our model outperform the existing state-of-the-art systems.

Conclusion

In this paper, we proposed a novel approach for tweet normalization and semantic similarity. The proposed method adopted a sequence to sequence model with attention-based mechanism, which is used for tweet normalization. After normalized text, the model with pre-trained is used for paraphrase identification and sematic similarity. Our model leverages subword information through a character-level convolutional neural network which output is used as an input to a recurrent neural network, and it is beneficial to morphologically rich languages with long-tailed frequency distribution or domains with dynamic vocabularies. A max-over-time pooling operation for convolutional neural networks is applied to obtain a fixed-deimensional representation of word, which is given to a highway network. The highway network's output is used as the input to long short-term memory. The encoder encodes a variable source sequence to a fixed dimensional representation and decoder decodes the representation to a variable target sequence. The end-to-end model normalizes informal text, and the pre-trained model is used for sentence semantic representation further. We test the effectiveness of our models in the paraphrase identification and semantic similarity in Twitter. Our attention-based character-level input method yields large gains over non-attention models for the paraphrase identification and semantic similarity. In the future, we plan to apply the proposed methods for question answer tasks in social media.

References

Bahdanau, D.; Cho, K.; and Bengio, Y. 2014. Neural machine translation by jointly learning to align and translate. *CoRR* abs/1409.0473.

Bertero, D., and Fung, P. 2015. Hltc-hkust: A neural network paraphrase classifier using translation metrics, semantic roles and lexical similarity features. In *Proceedings of SemEval*, 23–28. Denver, Colorado: Association for Computational Linguistics.

Bicici, E. 2015. Rtm-dcu: Predicting semantic similarity with referential translation machines. In *Proceedings of SemEval*, 56–63. Denver, Colorado: Association for Computational Linguistics.

Das, D., and Smith, N. A. 2009. Paraphrase identification as probabilistic quasi-synchronous recognition. In *Proceedings of ACL*, 468–476. Suntec, Singapore: Association for Computational Linguistics.

Ganitkevitch, J.; Van Durme, B.; and Callison-Burch, C. 2013. Ppdb: The paraphrase database. In *Proceedings of NAACL*, 758–764. Atlanta, Georgia: Association for Computational Linguistics.

Gulcehre, C.; Ahn, S.; Nallapati, R.; Zhou, B.; and Bengio, Y. 2016. Pointing the unknown words. In *Proceedings of ACL*, 140–149. Berlin, Germany: Association for Computational Linguistics.

Guo, W., and Diab, M. 2012. Modeling sentences in the latent space. In *Proceedings of ACL*, 864–872. Jeju Island, Korea: Association for Computational Linguistics.

Han, B., and Baldwin, T. 2011. Lexical normalisation of short text messages: Makn sens a #twitter. In *Proceedings of ACL*, 368–378. Portland, Oregon, USA: Association for Computational Linguistics.

Henderson, J., and Popa, D. 2016. A vector space for distributional semantics for entailment. In *Proceedings of ACL*, 2052–2062. Berlin, Germany: Association for Computational Linguistics.

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

Hochreiter, S. 1998. The vanishing gradient problem during learning recurrent neural nets and problem solutions. *Int. J. Uncertain. Fuzziness Knowl.-Based Syst.* 6(2):107–116.

Jan, A. B., and Blunsom, P. 2014. Compositional morphology forword representations and language modelling. In *Proceedings of ICML*, volume 32. Beijing, China: JMLR.

Kim, Y.; Jernite, Y.; Sontag, D.; and Rush, A. 2016.

Character-aware neural language models. In *Proceedings of AAAI*, 2741–2749.

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

Li, C., and Liu, Y. 2015. Joint pos tagging and text normalization for informal text. In *Proceedings of IJCAI*, IJCAI'15, 1263–1269. AAAI Press.

Liang, C., and Paritosh, P. 2016. Learning paraphrase identification with structural alignmen. In *Proceedings of IJCAI*, 2859–2865.

Ling, W.; Dyer, C.; Black, A. W.; and Trancoso, I. 2013. Paraphrasing 4 microblog normalization. In *Proceedings of EMNLP*, 73–84. Seattle, Washington, USA: Association for Computational Linguistics.

Luong, T.; Pham, H.; and Manning, C. D. 2015. Effective approaches to attention-based neural machine translation. In *Proceedings of EMNLP*, 1412–1421. Lisbon, Portugal: Association for Computational Linguistics.

Mikolov, T.; Karafiat, M.; Burget, L.; Cernock, J.; and Khudanpur, S. 2010. Recurrent neural network based language model.

Mueller, J., and Thyagarajan, A. 2016. Siamese recurrent architectures for learning sentence similarity. In *Proceedings of AAAI*, 2786–2792.

Pavlick, E.; Bos, J.; Nissim, M.; Beller, C.; Van Durme, B.; and Callison-Burch, C. 2015a. Adding semantics to data-driven paraphrasing. In *Proceedings of ACL*, 1512–1522. Beijing, China: Association for Computational Linguistics.

Pavlick, E.; Rastogi, P.; Ganitkevitch, J.; Van Durme, B.; and Callison-Burch, C. 2015b. Ppdb 2.0: Better paraphrase ranking, fine-grained entailment relations, word embeddings, and style classification. In *Proceedings of ACL*, 425–430. Beijing, China: Association for Computational Linguistics.

Pennell, D., and Liu, Y. 2011. A character-level machine translation approach for normalization of sms abbreviations. In *Proceedings of IJCNLP*, 974–982. Chiang Mai, Thailand: Asian Federation of Natural Language Processing.

Severyn, A., and Moschitti, A. 2015. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of SIGIR*, SIGIR '15, 373–382. ACM.

Socher, R.; Huang, E. H.; Pennin, J.; Manning, C. D.; and Ng, A. Y. 2011. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *Proceedings of NIPS*. Curran Associates, Inc. 801–809.

Sutskever, I.; Vinyals, O.; and Le, Q. V. 2014. Sequence to sequence learning with neural networks. In *Proceedings of NIPS*, NIPS'14, 3104–3112. Cambridge, MA, USA: MIT Press.

Tran, K.; Bisazza, A.; and Monz, C. 2016. Recurrent memory networks for language modeling. In *Proceedings of NAACL*, 321–331. San Diego, California: Association for Computational Linguistics.

van der Goot, R., and van Noord, G. 2015. Rob: Using semantic meaning to recognize paraphrases. In *Proceedings ofSemEval*, 40–44. Denver, Colorado: Association for Computational Linguistics.

Wang, M.; Lu, Z.; Li, H.; Jiang, W.; and Liu, Q. 2015. A convolutional architecture for word sequence prediction. In *Proceedings of ACL*, 1567–1576. Beijing, China: Association for Computational Linguistics.

Wieting, J.; Bansal, M.; Gimpel, K.; and Livescu, K. 2015. Towards universal paraphrastic sentence embeddings. *CoRR* abs/1511.08198.

Xu, W.; Ritter, A.; Callison-Burch, C.; Dolan, W. B.; and Ji, Y. 2014. Extracting lexically divergent paraphrases from Twitter. *Transactions of the Association for Computational Linguistics (TACL)* 2(1).

Xu, W.; Callison-Burch, C.; and Dolan, B. 2015. Semeval-2015 task 1: Paraphrase and semantic similarity in twitter (pit). In *Proceedings of SemEval*, 1–11. Denver, Colorado: Association for Computational Linguistics.

Yin, W., and Schütze, H. 2015. Convolutional neural network for paraphrase identification. In *Proceedings of NAACL*, 901–911. Denver, Colorado: Association for Computational Linguistics.

Zaremba, W., and Sutskever, I. 2014. Learning to execute. *CoRR* abs/1410.4615.

Zarrella, G.; Henderson, J.; Merkhofer, E. M.; and Strickhart, L. 2015. Mitre: Seven systems for semantic similarity in tweets. In *Proceedings of SemEval*, 12–17. Denver, Colorado: Association for Computational Linguistics.