# Tweet Modeling with LSTM Recurrent Neural Networks for Hashtag Recommendation

Jia Li, Hua Xu, Xingwei He, Junhui Deng and Xiaomin Sun
State Key Laboratory of Intelligent Technology and Systems
Tsinghua National Laboratory for Information Science and Technology
Department of Computer Science and Technology, Tsinghua University,
Beijing 100084, China

Email: callmecoding@sina.com, xuhua@tsinghua.edu.cn, hexingwei852@sina.com

***Abstract*—The hash symbol, called a hashtag, is used to mark the keyword or topic in a tweet. It was created organically by users as a way to categorize messages. Hashtags also provide valuable information for many research applications such as sentiment classification and topic analysis. However, only a small number of tweets are manually annotated. Therefore, an automatic hashtag recommendation method is needed to help users tag their new tweets. Previous methods mostly use conventional machine learning classifiers such as SVM or utilize collaborative filtering technique. A bottleneck of these approaches is that they all use the TF-IDF scheme to represent tweets and ignore the semantic information in tweets. In this paper, we also regard hashtag recommendation as a classification task but propose a novel recurrent neural network model to learn vector-based tweet representations to recommend hashtags. More precisely, we use a skip-gram model to generate distributed word representations and then apply a convolutional neural network to learn semantic sentence vectors. Afterwards, we make use of the sentence vectors to train a long short-term memory recurrent neural network (LSTM-RNN). We directly use the produced tweet vectors as features to classify hashtags without any feature engineering. Experiments on real world data from Twitter to recommend hashtags show that our proposed LSTM-RNN model outperforms state-of-the-art methods and LSTM unit also obtains the best performance compared to standard RNN and gated recurrent unit (GRU).**

## I. Introduction

In the past decade, Twitter has experienced tremendous success and has become one of the most important social network services. As the number of available tweets grows, the problem of managing tweets becomes extremely difficult. To avoid information overload, an efficient way called $hashtag$, the '#' symbol used before a relevant keyword or phrase, was introduced by Twitter to categorize tweets. Hashtags also attracted much attention in various research areas such as sentiment classification [1] and topic analysis [2], [3].

However, only 14.6% of tweets contain hashtags, since they should be manually annotated [4]. Therefore, a reliable hashtag recommendation method is needed to help users automatically tag their new tweets. To this end, many methods have been proposed to automatically recommend hashtags such as [5]–[9]. These methods either use conventional machine learning classifiers such as SVM or utilize collaborative filtering. Despite the differences in type of classifier used, these methods have a common weakness: they all use the TF-IDF scheme to represent tweets as features and ignore the semantic information in tweets. The frequent near-orthogonality [10] of TF-IDF greatly hinders the performance. In other words, the TF-IDF has become the bottleneck for hashtag recommendation. Recently Tomar et al. [11] utilize distributed word representations (word embeddings) and a feed forward neural network (FFNN) to recommend hashtags. Their method could recommend hashtags that are specific to the semantics of tweets. However, we think just use word embeddings to represent tweets is not enough.

We note that recurrent neural networks (RNNs) have obtained outstanding performance in representation learning field. Inspired by the recent improvement of document level sentiment classification, in this paper, we propose a LSTM-RNN model to learn semantic tweet representations for hashtag classification. Our method is based on the principle of compositionality [12], which states the meaning of a longer expression (e.g. a sentence) depends on the meanings of its units (e.g. a word). Specifically, our method models tweet representation in three steps: (a) it uses a skip-gram model with negative sampling (SGNS) to generate distributed word representations; (b) it utilizes a convolutional neural network (CNN) to compose sentence representations based on word embeddings; (c) it uses a long short-term memory (LSTM) model to encode the intrinsic (syntactic or semantic) relations between sentences into the tweet vectors. The tweet vectors are used as features to classify hashtag of each tweet. The details of our method is described in Section III.

We conduct sufficient experiments to compare our proposed method with state-of-the-art approaches: (1) Firstly, we compare our LSTM-RNN method with feed forward neural networks with distributed word representations and approaches based on TF-IDF include kNN, SVM, and Random Forests; (2) Secondly, to explore the effectiveness of tweet representations, we compare to LSTM-RNNs which only consider word embeddings; (3) Finally, we compare different types of recurrent

Hua Xu is corresponding author, Email:xuhua@tsinghua.edu.cn

978-1-5090-0620-5/16/$31.00 

units: standard RNN unit, LSTM unit, and gated recurrent unit (GRU). Our experiments reveal that: (1) tweet modeling with LSTM-RNNs outperforms state-of-the-art methods; (2) LSTM unit obtains the best performance among three RNN units for capturing tweet semantics.

The remainder of this paper is organized as follows. We briefly review related work in Section II. In Section III, we show the detail of our recurrent neural network method. Section IV presents the experimental results. We end this paper with conclusions and thoughts for future work in Section V.

## II. Related Work

***Hashtag Recommendation***: Hashtag recommendation in Twitter is a difficult and challenging problem because of the shortness and sparsity of tweets. Mazzia and Juett [5] apply a Naive Bayes model to classify tweets with hashtags and this method produces a ranked list of the top-20 recommended hashtags. Zangerle et al. [6] compare three approaches based on the TF-IDF representations of tweets to recommend hashtags. Kywe et al. [7] suggest hashtags by combining hashtags of similar users and similar tweets. The TF-IDF scheme is used again in computing similar tweets. Otsuka et al. [9] propose the HF-IHU ranking scheme, which is a variation of TF-IDF, that considers hashtag relevancy. Sedhai and Sum [8] formulate recommending hashtags as a learning to rank problem and use RankSVM to rank the candidate hashtags. Furthermore, this method only deals with tweets with URLs.

However, the frequent near-orthogonality [10] of TF-IDF greatly hinders the performance of those methods in practical use. In other words, the TF-IDF scheme has become the bottleneck for hashtag recommendation. Recently a handful of works focus on using topic models to discover underlying topic information of tweets. Godin et al. [13] first apply latent Dirichlet allocation (LDA) to automatically recommend keywords as hashtags. Ding et al. [14] assume that hashtags and tweets are talking about the same content but written in different languages. They propose a method by combining a topic model and a translation model to solve the problem. Although topic models can mine theme representations of tweets, that's not enough. Tomar et al. [11] use distributed word representations and a feed forward neural network to classify hashtags. This method does not require carefully designed feature engineering.

***RNNs for NLP***: RNNs are powerful models that have shown great promise in many natural language processing (NLP) tasks such as language modeling, question answering, sentiment classification, and machine translation. Here we briefly review two subareas: (1) *Language Modeling*. Mikolov et al. [15] present a new recurrent neural network based language model (RNNLM) with applications to speech recognition. The experimental results obtain 50% reduction of perplexity by using mixture of several RNNLMs. Mikolov et al. [16] propose several modifications of the original RNNLMs and these modifications outperform many competitive language modeling techniques in terms of accuracy. (2) *Machine Translation*. Kalchbrenner and Blunsom [17] introduce a class of probabilistic continuous machine translation models called recurrent continuous translation models that purely based on distributed representations for words, phrases, and sentences. Cho et al. [18] propose a novel neural network model called RNN Encoder-Decoder to learn phrase representations for machine translation. Luong et al. [19] explore useful architectures for attention-based RNN Encoder-Decoder models. With local attention, RNN Encoder-decoder models have achieved a significant gain of 5.9 BLEU points over non-attentional systems.

## III. The Proposed Approach

In this section, we describe the details of our proposed LSTM-RNNs for hashtag recommendation. The whole framework is displayed in Fig. 1. Our approach consists of four components: word embeddings generation, sentence composition, tweet composition, and hashtag classification.

Firstly, our approach generates word embeddings via a skip-gram model with negative sampling for each word in tweets (Section III.$A$). Then based on the compositionality of elements, which states the meaning of a longer expression (e.g. a sentence or a tweet) comes from the meanings of its components and the rules used to combine them. It is clear that a sentence consists of a list of words and a tweet consists of several sentences. In sentence composition, our approach uses a CNN to generate distributed sentence representations with word embeddings (Section III.$B$). In tweet composition, a LSTM-RNN model is used to produce vector-based tweet representations (Section III.$C$). Finally, the distributed tweet representations are directly regarded as features to classify hashtags (Section III.$D$).

### A. Word Embeddings Generation

Word embeddings, or distributed representations of words, have been the subject of much recent work in deep learning and natural language processing areas, demonstrating state-of-the-art performance on word analogy and word similarity tasks, amongst others. Word embeddings use dense, real-valued vectors to represent words from the vocabulary. Unlike one-hot vectors only indicate the location of a word in the vocabulary, word embeddings have much smaller dimension and carry syntactic and semantic information about the words. In this paper, we use the skip-gram model introduced in [20] to generate word embeddings. Fig. 2 shows the architecture of the skip-gram model which consists of three layers: an input layer, a projection layer, and an output layer.

Technically, each current word is used as an input to a log-linear classifier with continuous projection layer, and the classifier predicts words within a certain range before and after the current word. The range makes a trade-off between effectiveness and efficiency. The increasing range improves quality of word embeddings, but also leads to expensive computation.

The training complexity $Q$ of the skip-gram is proportional to

$$Q = C \times (D + D \times log_2(V)), \tag{1}$$

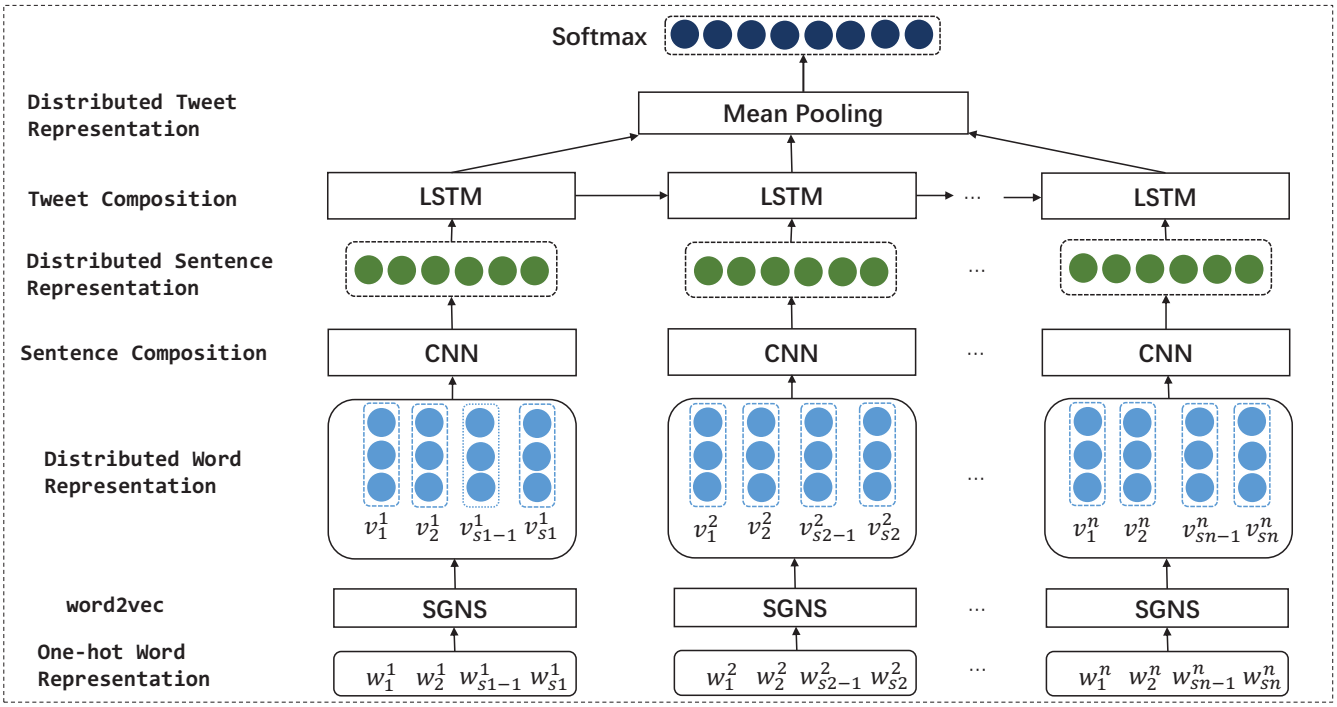



Fig. 1. The framework of tweet modeling for hashtag recommendation.

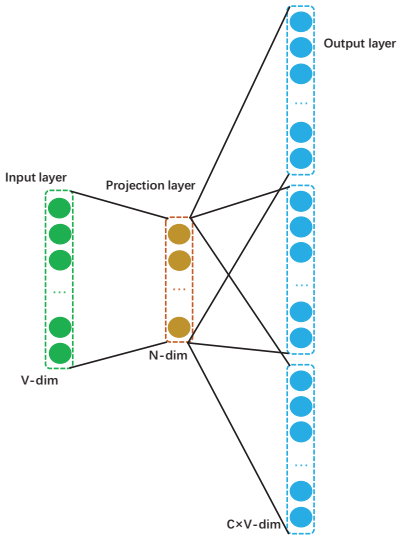



Fig. 2. The skip-gram model architecture.

where $C$ is the maximum distance between the words (namely, $C$ is the range of prediction), $D$ is the dimension of feature vectors, and $V$ is the size of the vocabulary.

In this paper, we also adopt negative sampling to optimize computational efficiency. More precisely, the negative sampling trick optimizes only the computation of the updates for output vectors. This idea is very straightforward: in order to solve the difficulty of there being too many times units to compute for each iteration, we can just sample some of the which we compute scores for. Mikolov et al. [20] have demonstrated the effectiveness of the skip-gram with negative sampling (SGNS) and SGNS outperforms other variants such as hierarchical softmax on analogy tasks.

### B. Sentence Composition

Recently a handful of studies use convolutional neural networks (CNNs) to model sentences such as [21]–[24] and obtain state-of-the-are performance in various NLP tasks. In this paper, we also apply a CNN model to compute vector-based sentence representations via semantic composition. The CNN model can learn a fixed-length vector from a variable-length input and captures the initial word order in a sentence.

The architecture of CNN model is shown in Fig. 3. The novelty of our CNN model is that the convolutional filters have different widths. Each filter contains a list of linear layers which also share weights and biases [24]. Applying multiple filters to capture local semantics of $n$-grams has been demonstrated effective in NLP tasks. Here, we use different widths in order to encode more local semantics about each single word. The widths we choose in this work are 1, 2 and 3 which refer to unigrams, bigrams and trigrams. Without loss of generality, we assume the current sentence is $\{v_1, v_2, ..., v_n\}$, where $v_i$ is the vector representation of word $w_i$. Let $l_f$ be the width of a filter, $W_f$ and $b_f$ be the shared parameters (that is, weights and biases) of linear layers in filter $f$. The input of a linear layer is $I_f = [v_i, v_{i+1}, ..., v_{i+l_f-1}] \in \mathbb{R}^{d \cdot l_f}$ and the output is:

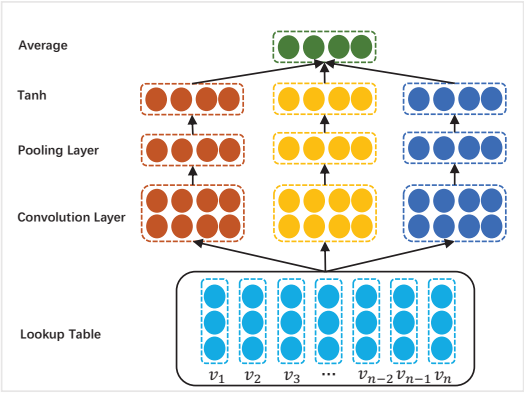


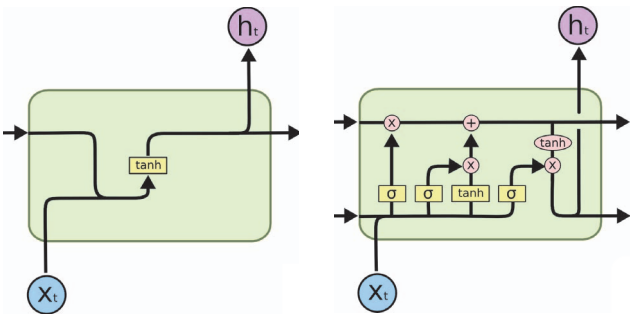

Fig. 3. The process of using a CNN model to compose sentence vectors.

$$Q_f = W_f \cdot I_f + b_f \tag{2}$$

where $W_f \in \mathbb{R}^{l_{of} \times d \cdot l_f}$, $b_f \in \mathbb{R}^{l_{of}}$, $l_{of}$ is the output length of the linear layer. Although convolutional layers could encode rich local semantics but they ignore the global information of each single word. We then add an *average* pooling layer to average the outputs of linear layers and obtain a fixed-length vector. Afterwards, we compute the *tanh* value of each vector and average the outputs of all filters to get the vector-based sentence representation.

### C. LSTM-RNN for Tweet Composition

Before introducing the tweet composition, we first review the long short-term memory recurrent neural network (LSTM-RNN).




(a) RNN (b) LSTM

Fig. 4. The internal structure of standard RNN unit (left) and LSTM (right)http://colah.github.io/posts/2015-08-Understanding-LSTMs/.

All RNNs have the form of a chain of repeating components of neural network. The repeating component in standard RNNs has a very simple structure, such as a single tanh layer (Fig. 4 (a)[1]). LSTM also has this chain like form, but its repeating module also called *memory cell* has a very different structure. Instead of having a single layer, there are four, interacting in a very special way (Fig. 4 (b)). LSTM was introduced by [25] primarily to solve the problem of vanishing gradients in standard RNNs.

The key to LSTM is the cell state, a horizontal line through the top of the *memory cell*. To clearly describe the intuition behind LSTM, we can imagine the cell state as a conveyor belt. It runs through the entire chain, with only minor linear interactions. Each *memory cell* could remove or add information to the cell state via carefully designed structures called gates. A gate consists of a sigmoid layer and a pointwise multiplication operation. The output value of sigmoid layer ranges from 0 to 1 which can be seen how much of information should be let through. Here, we take *memory cell* $c$ as an example to describe its detailed structure:

- *Forget gate*: The forget gate provides a forgetting coefficient by looking at the input layer $x_t$ and previous hidden layer $h_{t-1}$ for cell state $C_{t-1}$. The coefficient ranges from 0 to 1 and controls the information from $C_{t-1}$ to $C_t$.

$$f_t = \sigma(W_f \cdot [h_{t-1}, x_t] + b_f) \tag{3}$$

- *Input node*: This unit also takes activation from the input layer $x_t$ and previous hidden layer $h_{t-1}$. Typically, a tanh layer is used to process the summed weighted input.

$$g_t = tanh(W_g \cdot [h_{t-1}, x_t] + b_g) \tag{4}$$

- *Input gate*: The input gate decides which values should be updated in $C_{t-1}$ and its output multiplies the value of input node to get a new candidate of $C_t$.

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{5}$$

- *Internal state*: The heart of memory cell $c$ is the internal state: $C_t$.

$$C_t = f_t * C_{t-1} + i_t * g_t \tag{6}$$

- *Output gate*: The hidden layer $h_t$ is produced by the internal state $C_t$ and the value of the output gate $o_t$.

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{7}$$

$$h_t = o_t * tanh(C_t) \tag{8}$$

Since LSTM-RNNs are effective at capturing long-term memory dependencies without suffering from vanishing gradients, they have been used for many NLP tasks. In this paper, we novelly use LSTM-RNNs to learn semantic vector representations for tweets. The process of generating tweet representations is shown in Fig. 5.

Given a variable-length sequence of sentence vectors as input, LSTM-RNNs produce a fixed-length tweet vector. Unlike the recently proposed Encoder-Decoder models in neural machine translation, most of them just output last hidden state as the fixed-size vector [26]. We not only utilize the sequence summarization property of LSTM-RNNs but also consider the global semantics of each tweet. Here, we average *each* hidden state $h_t$ to produce the tweet vector.

More precisely, the process of generating a tweet vector works as follows. For tweet $d$, we first generate its sentence vectors $s\{x_1, x_2, ..., x_t, ..., x_n\}$ via the CNN model. Then

[1]http://colah.github.io/posts/2015-08-Understanding-LSTMs/

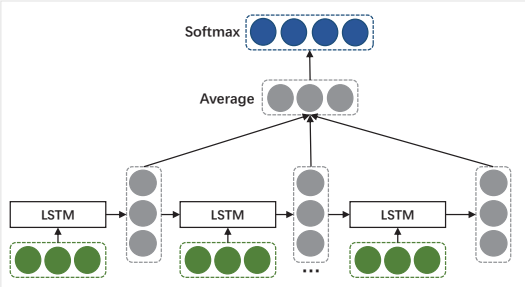



Fig. 5. The process of generating distributed tweet representations via LSTM-RNNs.

these sentence vectors are regarded as the input to the LSTM-RNN model. In each timestamp $t$, LSTM-RNN processes the input $x_t$ and previous cell state $C_{t-1}$ to generate current hidden state $h_t$ and output it. After getting all of the hidden states $\{h_1, h_2, ..., h_n\}$, we calculate their average and output it as $d's$ vector representation.

### D. Hashtag Classification

Previous papers mostly regard the TF-IDF representations as features of tweets. Instead, we consider the semantic tweet representations as features without other feature engineering. The tweet representation is fed to a softmax layer whose target is the class label (hashtag) associated with the input vector. Softmax function is a generalization of the logistic function that squashes a $K$-dimensional vector $x$ of arbitrary real values to a $K$-dimensional vector $\sigma(x)$ of real values in the range (0,1) that add up to 1.

$$\sigma(x)_j = \frac{e^{x_j}}{\sum_{k=1}^{K} e^{x_k}}, \quad for \;\; j = 1, 2, ..., K \tag{9}$$

In this work, we choose the cross-entropy error between real hashtag distribution $P(d)$ and predicted hashtag distribution $Q(d)$ as the cost function, although there are several other choices such as mean squared error.

$$f_{cost} = -\sum_{d \in D} \sum_{i=1}^{C} P_i(d) \cdot log(Q_i(d)) \tag{10}$$

where $D$ represents the training data, $d$ refers to a tweet, and $C$ is the number of classes. Note we use a one-hot $P(d)$ representation, which has a $C$ dimension and only the dimension corresponding to $d$'s hashtag is 1 with all others being 0.

## IV. Experiments

In this study, we conduct sufficient experiments to compare our proposed method with state-of-the-art approaches: (1) Firstly, we compare with feed forward neural networks with distributed word representations and approaches include kNN, SVM, and Random Forests; (2) Secondly, to explore the effectiveness of tweet representations, we compare to LSTM-RNNs which only consider word embeddings; (3) Finally, we compare different types of recurrent units: standard RNN unit, LSTM unit and gated recurrent unit (GRU).

TABLE I
Statistics of the classification dataset.

| | |
|---|---|
| Number of tweets | 42,108 |
| Number of users | 5,015 |
| Number of words | 69,431 |
| Number of hashtags | 20 |
| Time Range | 2015/06/01 $\sim$ 2015/08/31 |

### A. Dataset and Preprocess

We used the public Twitter streaming API[2] to collect 1,330,908 tweets from June 1 to August 31, 2015. Of these tweets, approximately 20.6% contain hashtags. After observing the raw data, we carefully designed a series of steps to process them. Firstly, using regular expressions to remove non-English tweets, which here were Chinese and Japanese tweets, and there were 1,210,812 pieces of tweet left. To get pure text contents, we removed retweets, URLs, punctuations, mentions, and HTML entities. Then we applied an English stop word list to remove meaningless stopwords. Since we use tweets and hashtags for training our LSTM-RNN model, tweets without hashtags are not useful. Consequently, we removed tweets without hashtags. Then we selected the top 20 popular hashtags and the tweets containing the popular hashtags. There were 42,000 tweets left. Finally, We split the pre-processed tweets into a training dataset (80%), a validation dataset (10%), and a test dataset (10%).

It's clear that one tweet may contain several hashtags and there may be different hashtags to describe one event. Besides the top 20 hashtags, we also extract 5 similar other hashtags for each hashtag.

### B. Evaluation Metric

As far as we know, there isn't a best evaluation metric for hashtag recommendation. In our experiments, we compared the performance of different approaches by two evaluation methods: accuracy and hitrate. $Accuracy$ is the proportion of true results among the total number of cases examined. It is commonly used in machine learning classification tasks. When using accuracy as the criteria, all of methods only suggest one hashtag for each tweet in test sets.

$Hitrate$ has been used in most previous work such as [13], [27], [28]. In order to recommend several hashtags rather than only one hashtag, we used the extracted hashtags to enlarge our training dataset by 5 times. Technically, we replaced the original hashtag by the similar hashtag. We ask 3 volunteers (represented by $p_1, p_2, p_3$) to independently hit one recommended hashtag if it could possibly be a hashtag for the test tweet. However, there may be different hashtags to describe one concept. Therefore, we ask the participants to hit a suggeseted hashtag as if its meaning is similar to

[2]https://dev.twitter.com/streaming/overview

the original hashtag. We define the $hitrate_{p_i}@N$: given $N$ suggested hashtags for each test tweet, if $p_i$ think one could be used as a hashtag, he would hit it. Hence,

$$hitrate@N = \frac{average\ number\ of\ hitting\ tweets}{number\ of\ test\ tweets}. \quad (11)$$

where N$\in [5, 10]$.

### C. Compared Methods and Implementation

We compare our LSTM-RNN model with several state-of-the-art hashtag classification approaches described briefly as follows:

- ***kNN***: The kNN method classifies a new tweet $d_1$ via a similarity measure. Here, we use the cosine distance as the similarity measure. That is, kNN computes the distance between $d_1$ and another tweet $d_2$ from the training set:

$$cos(d_1, d_2) = \frac{d_1 \cdot d_2}{||d_1|| \cdot ||d_2||} \quad (12)$$

  The features were generated by computing the TF-IDF scheme of each tweet. We set k$\in\{50,100\}$. The kNN classifier was trained by using the scikit-learn toolkit ($sklearn$) [29].
- ***SVM***: The SVM algorithm is well known for its very good practical results. It's also a powerful classifier for hashtag classification. We made use of the libsvm [30] to train SVM: (1) SVM-Linear with linear kernel; (2) SVM-RBF with rbf kernel. Like kNN, we also used the TF-IDF representations as the tweet features.
- ***Random Forests***: The Random Forests algorithm is an ensemble learning method for classification and other tasks, that operates by constructing a multitude of decision trees and outputting the class that is the mode of the classes of the individual trees. The number of decision trees we set was 50. We also made use of sklearn to train Random Forests. The TF-IDF scheme was used again to generate Random Forests input.
- ***Feed Forward Neural Network (FFNN)***: A FFNN is an artificial neural network where connections between the units do not form a cycle. FFNNs are widely used in many practical applications, and [11] first uses FFNNs to classify hashtags with distributed word representations. We implemented a three ReLU layers FFNN via Caffe [31] and used the word embeddings as input.
- ***LSTM-RNN***: We made use of Caffe to implement a two layers LSTM-RNN model with 200 cells at each layer. Afterwards, we got two LSTM-RNN models: LSTM-word took word embeddings as input, and LSTM-tweet considered tweet vectors as input.
- ***RNN***: We implemented a two layers RNN with Caffe which considered the tweet vectors as input called RNN-tweet.
- ***Gated Recurrent Unit (GRU)***: GRU is another sophisticated recurrent unit proposed in [32]. Recently [33] empirically evaluated GRU on some sequence modeling tasks and found GRU to be comparable to LSTM. We also used Caffe to implement a two layers GRU-tweet which considered the tweet vectors as input.

It is important to note that our method is supervised. Therefore we do not compare with any unsupervised methods such as LDA [13]. The training details about neural networks are given below: (1) We used the gensim[3] [34] Python implementation of Word2Vec to train a word embedding model. Inspired by Lai et al. [35] that more training data would be beneficial to train a precise model, we combined the full English wikipedia[4] and the New York Times corpus[5] to obtain the gensim training corpus. Each word has a 300-dimensional feature vector. (2) LSTM parameters were initialized from a uniform distribution between [-0.05,0.05]. For CNN, we initialized the weights in each layer from a zero-mean Gaussian distribution with standard deviation 0.01. The biases were initialized with the constant 0.1. (3) All of the neural networks were trained using momentum-accelerated mini-batch SGD and momentum set to 0.9. (4) The batch size of RNN was set to 32 (32 tweets).

### D. Experimental Results

**Accuracy Performance Comparison** Table II lists the accuracies of all the approaches on our test set. We have the following experimental results: (1) LSTM-tweet achieves the highest accuracy based on distributed tweet representations and LSTMs. This demonstrates that our proposed method can suggest more accurate hashtags; (2) GRU-tweet falls behind LSTM-tweet but performs better than RNN-tweet. This is because GRUs can capture more semantics and have better capacity to summarize tweet information than standard RNNs; (3) Neural Network methods basically perform better than kNN, SVM and Random Forests. One reason may be the input for neural networks is word embeddings or tweet vectors rather than TF-IDF forms of tweets. The distributed word or tweet representations contain more semantic features about words. In addition, we think the great ability to generalizaion and response to unexpected patterns of neural networks is another reason; (4) Neural Networks based on tweet vectors outperform those based on word vectors. A reasonable explanation is that the effectiveness of our tweet modeling with LSTMs and sentence composition with CNNs.

**Hitrate Performance Comparison** Table III shows the overall hitrate results. As shown by the highest hitrates in bold type, it is obvious that LSTM-tweet basically takes the lead. Compared to RNN unit and GRU unit, LSTM unit is a fine choice for hashtag recommendation. Fig. 6 also illustrates the results of standard RNN, GRU and LSTM. In this study, we not only novelly use LSTM-RNNs to classify hashtags but also propose a framework of modeling tweets based on the principle of compositionality. Experiments show that tweet vectors are more suitable as input for neural networks than

[3]https://radimrehurek.com/gensim/

[4]https://dumps.wikimedia.org/enwiki/latest/enwiki-latest-pages-articles.xml.bz2

[5]https://catalog.ldc.upenn.edu/LDC2008T19

TABLE III
THE HITRATES (HIGHER IS BETTER) OF ALL METHODS FOR HASHTAG SUGGESTION. THE BEST METHOD IS IN **BOLD**.

| Model | Hitrate@5 | Hitrate@6 | Hitrate@7 | Hitrate@8 | Hitrate@9 | Hitrate@10 |
|---|---|---|---|---|---|---|
| FFNN | 65.3 | 66.4 | 67.1 | 67.1 | 68.6 | 69.5 |
| LSTM-word | 71.2 | 73.1 | 74.3 | 75.6 | 76.3 | 77.9 |
| RNN-tweet | 72.5 | 74.2 | 75.6 | 77.8 | 80.2 | 81.4 |
| GRU-tweet | 76.9 | **80.2** | 82.6 | 83.9 | 84.9 | 85.9 |
| LSTM-tweet | **77.6** | 79.3 | **82.7** | **84.1** | **85.5** | **86.4** |

TABLE II
ACCURACY RESULTS (HIGHER IS BETTER) FOR HASHTAG CLASSIFICATION. THE BEST METHOD IS IN **BOLD**.

| Method | Accuracy |
|---|---|
| kNN-50 | 19.4 |
| kNN-100 | 22.1 |
| SVM-Linear | 22.7 |
| SVM-RBF | 21.3 |
| Random Forests | 24.7 |
| FFNN | 24.2 |
| LSTM-word | 25.9 |
| RNN-tweet | 26.3 |
| GRU-tweet | 27.7 |
| LSTM-tweet | **28.6** |

word embeddings. This is because tweet vectors contain word features, local relations of words, and global semantics of sentences. Without any feature engineering, our end-to-end method achieves the best performance and could recommend suitable hashtags for new tweets.

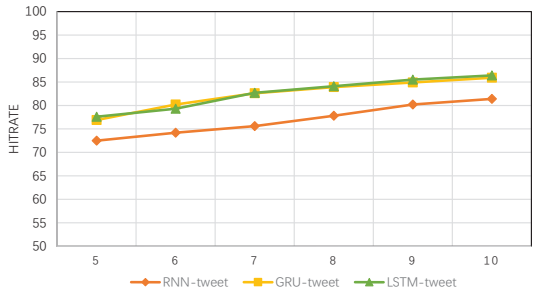



Fig. 6. The hitrates of three RNN units: standard RNN unit, LSTM unit, and GRU unit.

## V. CONCLUSION AND FUTURE WORK

In this paper, we introduce neural networks (CNNs and LSTM-RNNs) for tweet modeling to recommend hashtags in Twitter. Our proposed distributed tweet representations not only encode word features but also contain semantics of sentences and sentence relations. The method to suggest hashtags consists of four components: word embeddings generation, sentence composition, tweet composition, and hashtag classification. The experimental results show that our LSTM-RNNs can outperform other state-of-the-art supervised methods such as SVM, Random Forests and FFNNs for hashtag recommendation. We also evaluate three commonly used RNN units: standard RNN, GRU and LSTM. LSTM unit achieves the best performance in our dataset.

As future research, we also plan to study the effectiveness of sentence composition via RNNs. Another future direction lies in incorporating other attributes of tweets such as authors, mentions, and URLs in our method.

## ACKNOWLEDGMENTS

This work is supported by National Basic Research Program of China (973 Program) (Grant No: 2012CB316301).

## REFERENCES

[1] D. Davidov, O. Tsur, and A. Rappoport, "Enhanced sentiment learning using twitter hashtags and smileys," in *Proceedings of the 23rd International Conference on Computational Linguistics: Posters*. Association for Computational Linguistics, 2010, pp. 241–249.

[2] A. Cui, M. Zhang, Y. Liu, S. Ma, and K. Zhang, "Discover breaking events with popular hashtags in twitter," in *Proceedings of the 21st ACM international conference on Information and knowledge management*. ACM, 2012, pp. 1794–1798.

[3] X. Meng, F. Wei, X. Liu, M. Zhou, S. Li, and H. Wang, "Entity-centric topic-oriented opinion summarization in twitter," in *Proceedings of the 18th ACM SIGKDD international conference on Knowledge discovery and data mining*. ACM, 2012, pp. 379–387.

[4] X. Wang, F. Wei, X. Liu, M. Zhou, and M. Zhang, "Topic sentiment analysis in twitter: a graph-based hashtag sentiment classification approach," in *Proceedings of the 20th ACM international conference on Information and knowledge management*. ACM, 2011, pp. 1031–1040.

[5] A. Mazzia and J. Juett, "Suggesting hashtags on twitter," *EECS 545m, Machine Learning, Computer Science and Engineering, University of Michigan*, 2009.

[6] E. Zangerle, W. Gassler, and G. Specht, "Recommending#-tags in twitter," in *Proceedings of the Workshop on Semantic Adaptive Social Web (SASWeb 2011). CEUR Workshop Proceedings*, vol. 730, 2011, pp. 67–78.

[7] S. M. Kywe, T.-A. Hoang, E.-P. Lim, and F. Zhu, "On recommending hashtags in twitter networks," in *Social Informatics*. Springer, 2012, pp. 337–350.

[8] S. Sedhai and A. Sun, "Hashtag recommendation for hyperlinked tweets," in *Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval*. ACM, 2014, pp. 831–834.

[9] E. Otsuka, S. A. Wallace, and D. Chiu, "Design and evaluation of a twitter hashtag recommendation system," in *Proceedings of the 18th International Database Engineering & Applications Symposium*. ACM, 2014, pp. 330–333.

[10] D. Greene and P. Cunningham, "Practical solutions to the problem of diagonal dominance in kernel document clustering," in *Proceedings of the 23rd international conference on Machine learning*. ACM, 2006, pp. 377–384.

[11] A. Tomar, F. Godin, B. Vandersmissen, W. De Neve, and R. Van de Walle, "Towards twitter hashtag recommendation using distributed word representations and a deep feed forward neural network," in *Advances in Computing, Communications and Informatics (ICACCI, 2014 International Conference on*. IEEE, 2014, pp. 362–368.

[12] F. J. Pelletier, "The principle of semantic compositionality," *Topoi*, vol. 13, no. 1, pp. 11–24, 1994.

[13] F. Godin, V. Slavkovikj, W. De Neve, B. Schrauwen, and R. Van de Walle, "Using topic models for twitter hashtag recommendation," in *Proceedings of the 22nd international conference on World Wide Web companion*. International World Wide Web Conferences Steering Committee, 2013, pp. 593–596.

[14] Z. Ding, X. Qiu, Q. Zhang, and X. Huang, "Learning topical translation model for microblog hashtag suggestion," in *Proceedings of the Twenty-Third international joint conference on Artificial Intelligence*. AAAI Press, 2013, pp. 2078–2084.

[15] T. Mikolov, M. Karafiát, L. Burget, J. Cernockỳ, and S. Khudanpur, "Recurrent neural network based language model." in *INTERSPEECH 2010, 11th Annual Conference of the International Speech Communication Association, Makuhari, Chiba, Japan, September 26-30, 2010*, 2010, pp. 1045–1048.

[16] T. Mikolov, S. Kombrink, L. Burget, J. H. Černockỳ, and S. Khudanpur, "Extensions of recurrent neural network language model," in *Acoustics, Speech and Signal Processing (ICASSP), 2011 IEEE International Conference on*. IEEE, 2011, pp. 5528–5531.

[17] N. Kalchbrenner and P. Blunsom, "Recurrent continuous translation models." in *EMNLP*, 2013, pp. 1700–1709.

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

[19] M.-T. Luong, H. Pham, and C. D. Manning, "Effective approaches to attention-based neural machine translation," *arXiv preprint arXiv:1508.04025*, 2015.

[20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States.*, 2013, pp. 3111–3119. [Online]. Available: http://papers.nips.cc/paper/5021-distributed-representations-of-words-and-phrases-and-their-compositionality

[21] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," *arXiv preprint arXiv:1404.2188*, 2014.

[22] Y. Kim, "Convolutional neural networks for sentence classification," *arXiv preprint arXiv:1408.5882*, 2014.

[23] B. Hu, Z. Lu, H. Li, and Q. Chen, "Convolutional neural network architectures for matching natural language sentences," in *Advances in Neural Information Processing Systems*, 2014, pp. 2042–2050.

[24] D. Tang, B. Qin, and T. Liu, "Learning semantic representations of users and products for document level sentiment classification," in *Proc. ACL*, 2015.

[25] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

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

[27] J. She and L. Chen, "Tomoha: Topic model-based hashtag recommendation on twitter," in *Proceedings of the companion publication of the 23rd international conference on World wide web companion*. International World Wide Web Conferences Steering Committee, 2014, pp. 371–372.

[28] S. M. Kywe, T.-A. Hoang, E.-P. Lim, and F. Zhu, "On recommending hashtags in twitter networks," in *Social Informatics*. Springer, 2012, pp. 337–350.

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

[30] C.-C. Chang and C.-J. Lin, "Libsvm: A library for support vector machines," *ACM Transactions on Intelligent Systems and Technology (TIST)*, vol. 2, no. 3, p. 27, 2011.

[31] Y. Jia, E. Shelhamer, J. Donahue, S. Karayev, J. Long, R. Girshick, S. Guadarrama, and T. Darrell, "Caffe: Convolutional architecture for fast feature embedding," in *Proceedings of the ACM International Conference on Multimedia*. ACM, 2014, pp. 675–678.

[32] K. Cho, B. van Merriënboer, D. Bahdanau, and Y. Bengio, "On the properties of neural machine translation: Encoder-decoder approaches," *arXiv preprint arXiv:1409.1259*, 2014.

[33] J. Chung, C. Gulcehre, K. Cho, and Y. Bengio, "Empirical evaluation of gated recurrent neural networks on sequence modeling," *arXiv preprint arXiv:1412.3555*, 2014.

[34] R. Řehůřek and P. Sojka, "Software framework for topic modelling with large corpora," in *Proc. LREC Workshop on New Challenges for NLP Frameworks*, 2010, pp. 46–50.

[35] S. Lai, K. Liu, L. Xu, and J. Zhao, "How to generate a good word embedding?" *CoRR*, vol. abs/1507.05523, 2015. [Online]. Available: http://arxiv.org/abs/1507.05523