Model the Long-term Post History for Hashtag Recommendation*

Minlong Peng, Qiyuan Bian, Qi Zhang, Tao Gui, Jinlan Fu, Lanjun Zeng, and Xuanjing Huang

School of Computer Science, Fudan University, Shanghai, China
{mlpeng16,qybian19,qz,tgui16,fujl16,ljzeng18,xjhuang}@fudan.edu.cn

Abstract. The goal of this work is to provide a keyword-suggestion-like hashtag recommendation service, which recommends several hashtags when the user types in the hashtag symbol "#" while writing a post. Different from previously published hashtag recommendation systems, which only considered the textual information of the post itself or a few numbers of the latest posts, this work proposed to model the long-term post history for the recommendation. To achieve this purpose, we organized the historical posts of a user in the time order, obtaining a post sequence. Based on this sequence, we proposed a recurrent-neural-network-based framework, called the Parallel Long Short-term Memory (PLSTM), to perform the post history modeling. This was motivated by the success of the recurrent neural network in modeling the long-term dependency of dynamic sequences. The hashtag recommendation was performed based on both the current post content representation and the post history representation. We evaluated the proposed model on a real dataset crawled from Twitter. The experimental results demonstrated the effectiveness of our proposed model. Moreover, we quantitatively studied the informativeness of different parts of the post history and proved the feasibility of organizing the historical posts of a user in the time order.

Keywords: Hashtag recommendation · long-term post history · neural memory network.

1 Introduction

A hashtag (single token starting with a # symbol) is used to index keywords or topics on microblog services. It usually consists of natural language n-grams or abbreviations, e.g., "#Universe", "#MentalHealth", or "#US". People use the hashtag anywhere in their posts to indicate an object concisely or categorize those posts for easier searching. However, because there are an increasing number of hashtags, it is not easy for users to find an appropriate hashtag matching their intention when they intend to insert one. Therefore, it is necessary to provide a keyword-suggestion-like service that recommends several hashtags when a user wants to insert a hashtag (types in the # symbol).

In recent years, a variety of methods have been proposed from different perspectives to perform hashtag recommendation [1–5]. These approaches can be generally organized into two groups. The first group of methods mainly focus on modeling the

* The authors wish to thank the anonymous reviewers for their helpful comments.

textual content of posts, which is traditionally dominated by topic-model-based methods [6, 7]. Recently, this dominance has been overturned by a resurgence of interest in deep neural network (DNN) based approaches [8, 9]. The second group of methods additionally models the personal information of the user, which is commonly extracted from his/her post history. For example, Zhang et al., [10] proposed the TPLDA model for this purpose. It grouped posts by user and introduced an additional parameter for each user to the LDA model. The experiment results of these works have demonstrated the informativeness of the post history. However, most of these methods can only model a short and fixed length of post history. Thus, they in common truncated the post history and only modeled the latest, or the short-term, post history while ignoring the long-term post history for the recommendation.

In this work, we argue the informativeness of the long-term post history for the hashtag recommendation. One of the typical challenge in modeling the long-term post history results from its incrementally increasing size. It is either computationally unacceptable (TPLDA) or structurally inapplicable (HMemN2N) for the previous methods to model the long-term post history. For example, simply extending the memory size of HMemN2N to encode more historical posts will greatly increase the computation cost while even harm the performance. This results from two reasons. First, increasing the size of the post history will disturb their assumption that historical posts at different time step are equally important for the current recommendation. Second, it will greatly increases the input dimension of the recommendation module, making it easy over-fit the training data set. To tackle this challenge, we proposed a recurrent-neural-network-based model, called the Parallel Long Short-term Memory (PLSTM), to perform this task. It organized the historical posts of the user in the time order (demonstrated to be helpful in our experiments), resulting a post sequence, and then applies the PLSTM model to this sequence. This takes the advantage of the RNN in modeling the dynamic sequence and long-term dependency. In addition, it treats the content and hashtag as two different views of the post and models them separately, rather than treating the hashtags as normal words. We argue that this has several benefits. First, it does not need to constrain the representations of words and hashtags in the same vector space. Second, it reduces the word vocabulary size for representing the hashtags. Finally, it highlights the hashtag information.

In summary, the contributions of this paper are as follows: (**i**) We provide a keyword-suggestion-like hashtag recommendation service, which recommends several hashtags for choice when the user types in the "#" symbol. (**ii**) We model the entire post history for the recommendation using a recurrent-neural-network-based model. (**iii**) We quantitatively study the influence of different parts of the post history on the system performance and prove the feasibility of organizing posts in the time order.

2 Related Work

Hashtag recommendation has been extensively explored and developed over the last few years. Many approaches have been proposed from different perspectives to perform this task. In general, these approaches can be organized into two groups.

The first group of methods treats posts of different users without any distinction. It mainly focuses on modeling the post content. This is historically dominated by the topic-based-methods [6, 12–16]. Godin et al., [15] proposed to incorporate topic models to learn the underlying topic assignment of language classified tweets, and suggested hashtags for a tweet based on the topic distribution. Under the assumption that hashtags and tweets are parallel description of a resource, Ding et al. [14] tried to integrate latent topical information into translation model. However, with the development of neural networks, the dominance of topic-based approaches has recently been overturned by the neural-network-based approaches[8, 9, 17–19]. Dhingra et al., [8] treated a post as a character sequence and modeled it with a Long Short-term memory network. Gong and Zhang, [9] applied a convolution neural network to model the post content. They also introduced an attention mechanism into their system to select key features within the tweet.

The second group of methods also consider personal information of users when performing recommendations. Most of them extract the personal information from the post histories of users [7, 9, 11, 16]. Wang et al., [20] proposed combining the topic model with collaborative filtering. They extracted the user representation from the post history and predicted users' hashtag usage preferences in a collaborative filtering manner. Zhang et al., [10] grouped posts by user and introduced an additional parameter for each user in the LDA model. Huang et al., [11] constructed the post history with the latest five historical posts and stored them in an end-to-end memory. For recommendations, they recursively accessed the memory with the current post content. To deal with the dynamical length of the post history, the above method truncated the post history into a fixed length, with the latest historical posts left. In this work, however, we propose a recurrent framework to model the post history. It can model a dynamic number of historical posts and keep the long-term post history available for the current recommendation. Experimental results empirically demonstrated the effectiveness of this framework and proved the informativeness of the long-term post history.

3 Recurrent Hashtag Recommendation

The proposed model conceptually consists of four modules, i.e., the content representation module for encoding the post content, the hashtag representation module to obtain the hashtag representation, the recurrent module for modeling the post history, and the recommendation module. The general architecture is depicted in Figure 1. This figure was especially designed to highlight the core of this model in modeling the long-term post history for the recommendation.

3.1 Content Representation

In this work, we use a *one-layer convolution network* to model the post content. It is a variant of the traditional convolution network proposed by Kim et al., [21] for sentence encoding. And this architecture has been demonstrated to be quite effective for this task [9]. Specifically, let $\mathbf{w}_i \in R^{k_w}$ be the k_w-dimensional word vector, corresponding to

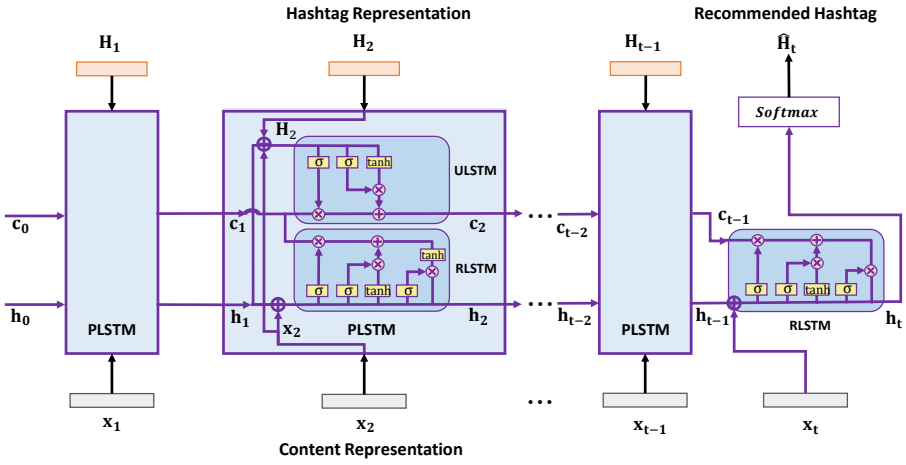

Fig. 1: General architecture of the recurrent hashtag recommendation framework. Here, $\mathbf{x}_i$ and $\mathbf{H}_i$ are the content and hashtag representation of the ith post. For the recommendation of a user at time step t, it first organizes the historical posts of the user in the time order, obtaining a content representation sequence $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{t-1}]$ and a hashtag representation sequence $[\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_{t-1}]$. Then, it models the post history with our proposed PLSTM recurrent network and obtains two vector representations of the post history $\mathbf{c}_{t-1}$ and $\mathbf{h}_{t-1}$. Finally, it performs the hashtag recommendation based on the mixed representation $\mathbf{h}_t$ of the post history and the current post content $\mathbf{x}_t$.

the i^{th} word of the post. A post of length n (padded if necessary) is represented as

$$\mathbf{p} = \left[\mathbf{w}_1, \cdots \mathbf{w}_n\right]$$

The *one-layer convolution network* takes the dot product of the filter $\mathbf{m} \in R^{k_w \times h}$ with each h-gram in $\mathbf{p}$ to obtain sequence $\mathbf{s}$ with the following:

$$\mathbf{s}_i = f(\mathbf{m} \cdot \mathbf{p}_{i:i+h-1} + b). \tag{1}$$

Here, $b \in R$ is a bias term, and f is the hyperbolic tangent (tanh) non-linear function. This filter is applied to each possible window of words in the sequence $\{\mathbf{p}_{1:h}, \mathbf{p}_{2:h+1}, \cdots, \mathbf{p}_{n-h+1:n}\}$ to produce a feature map:

$$\mathbf{s} = [\mathbf{s}_1, \cdots, \mathbf{s}_{n-h+1}]$$

To address the problem of various post lengths, it then applies a max-over-time pooling over the feature map and takes the maximum value $\hat{s} = \max(\mathbf{s})$ as the feature corresponding to this particular filter. By extending the operation to multiple filters with various window sizes, it obtains multiple features:

$$\mathbf{x} = \left[\max(\mathbf{s}^1) \cdots \max(\mathbf{s}^d)\right]. \tag{2}$$

Here d is the filter number and $\mathbf{s}^i$ denotes the feature map extracted with the i^{th} filter. These features form the representation of the post.

3.2 Hashtag Representation

Most of the previous hashtag recommendation systems just treat the hashtag as a single label and represent it with a randomly initialized trainable dense vector. This practice has two drawbacks. First, it loses the textual information of hashtags. Second, in this practice, the size of the hashtag set is fixed. Adding new hashtags is not possible. Therefore, in this work, we apply a recurrent neural network to the character sequence of hashtags to obtain their vector representations. The formal definition is precisely specified as follows:

$$\mathbf{h}_t = \tanh(\mathbf{W}_c \cdot \mathbf{c}_t + \mathbf{b}_c) \tag{3}$$

where $\mathbf{c}_t \in R^{d_c}$ is the vector representation of the t^{th} character of the hashtag $\mathbf{H}_i = \{\mathbf{c}_1, \mathbf{c}_2, \cdots, \mathbf{c}_n\}$, $\mathrm{W_c} \in \mathbb{R}^{d_p \times d_c}$ and $\mathrm{b_c} \in \mathbb{R}^{d_p}$ are trainable parameters of affine transformations. We use the final hidden state $\mathbf{h}_n \in \mathbf{R}^{d_p}$ to represent $\mathbf{H}_i$. And in the following, we refer $\mathbf{H}_i$ to this vector representation if without further explanation.

3.3 Model the Post History with Recurrent Neural Network

The general architecture of the framework is depicted in Figure 1. For the recommendation of a user at time step t, it organizes his/her historical posts in the time order. And because we model the post content and the used hashtag separately, we obtain two sequence representations of the post history, i.e., the content sequence $[\mathbf{x}_1, \mathbf{x}_2, \cdots, \mathbf{x}_{t-1}]$ and the hashtag sequence $[\mathbf{H}_1, \mathbf{H}_2, \cdots, \mathbf{H}_{t-1}]$. We then applied a recurrent framework PLSTM to the resulting two sequences, obtaining vector representations of the post history $\mathbf{c}_{t-1}$ and $\mathbf{h}_{t-1}$. Finally, we perform the hashtag recommendation based on the mixed representation $\mathbf{h}_t$ of the post history and the current post content. To achieve this purpose, we extend the long short-term memory network and design a parallel long short-term memory framework to perform this task which we describe below.

Parallel Long Short-term Memory The proposed recurrent module PLSTM contains two parallel LSTMs, with one (RLSTM) performing recommendations based on the memory and post content, and the other (ULSTM) performing memory updating. These two parallel LSTMs share the memory content as depicted in Figure 1. We separate the operations of recommendation and memory updating to make it easy to encode the object hashtag, chosen by the user after the recommendation performed, to the memory. The formal definition is as follows:

$$\begin{aligned}
\begin{bmatrix} \mathbf{i}_t \\ \mathbf{f}_t \\ \mathbf{o}_t \\ \widetilde{\mathbf{c}}_t \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}_r \begin{bmatrix} \mathbf{x}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b}_r \right), \\
\begin{bmatrix} \hat{\mathbf{i}}_t \\ \hat{\mathbf{f}}_t \\ \hat{\mathbf{c}}_t \end{bmatrix} &= \begin{bmatrix} \sigma \\ \sigma \\ \tanh \end{bmatrix} \left(\mathbf{W}_u \begin{bmatrix} \mathbf{x}_t \\ \mathbf{H}_t \\ \mathbf{h}_{t-1} \end{bmatrix} + \mathbf{b}_u \right), \\
\mathbf{c}_t &= \hat{\mathbf{c}}_t \odot \hat{\mathbf{i}}_t + \mathbf{c}_{t-1} \odot \hat{\mathbf{f}}_t, \\
\mathbf{h}_t &= \mathbf{o}_t \odot \tanh(\widetilde{\mathbf{c}}_t \odot \mathbf{i}_t + \mathbf{c}_{t-1} \odot \mathbf{f}_t).
\end{aligned} \tag{4}$$

Table 1: Statistics of the dataset.

Item	Train	Develop	Test
#User	2,000	1,000	1,000
#Tweet	127,846	8,086	9,190
#Example	187,247	13,174	13,946
#Hashtag	3,104	1,936	1,952
#Hashtag/User	23.54	6.28	6.29
#Word/Example	13.30	13.26	13.04

where σ is the element-wise sigmoid function and $\odot$ represents element-wise product. $\mathrm{W_r} \in \mathbb{R}^{4d_h \times (d_h+d_x)}$ and $\mathrm{b_r} \in \mathbb{R}^{4d_h}$ are trainable parameters for transformation. And $\mathrm{W_u} \in \mathbb{R}^{3d_h \times (d_h+d_x+d_p)}$ and $\mathrm{b_u} \in \mathbb{R}^{3d_h}$ are trainable parameters for memory updating. In addition, we argue that the recommendation and memory updating can be performed incrementally. This is achieved by feeding the object hashtag $\mathbf{H}_t$ into the recurrent module, updating $\mathbf{c}_{t-1}$ to $\mathbf{c}_t$. Note that $\mathbf{H}_t$ does not affect the recommendation of $\hat{\mathbf{H}}_t$. It only affects the recommendation for future posts $\mathbf{p}_{>t}$. This is reasonable and practicable because users will offer feedbacks immediately after the recommendation being performed. Specifically, once we have performed the recommendation, a user will immediately choose or type in a hashtag matching their intention. Thus, we can make use of the object hashtag to adjust our system accordingly for future recommendations.

3.4 Recommendation

the hashtag recommendation of the proposed model is performed based on the mixed representation $\mathbf{h}_t$ of the post history and the current post content. To obtain the rank of hashtag $\mathbf{H}_i$ as the recommendation candidate for post $\mathbf{p}_t$, we compare its representation with $\mathbf{h}_t$, obtaining a matching score:

$$\mathrm{score}(\mathbf{p}_t, \mathbf{H}_i) = \mathbf{h}_t^{\mathrm{T}} \mathbf{H}_i, \tag{5}$$

where $\mathbf{h}_t^{\mathrm{T}}$ denotes the transpose of $\mathbf{h}_t$. We apply a softmax non-linear operation to obtain the probability of it as the recommendation candidate, as follows:

$$p(\mathbf{H}_i|\mathbf{p}_t) = \frac{\exp\left(\mathrm{score}(\mathbf{p}_t, \mathbf{H}_i)\right)}{\sum_{j=0}^{\mathrm{N_H}} \exp\left(\mathrm{score}(\mathbf{p}_t, \mathbf{H}_j)\right)}, \tag{6}$$

where $\mathrm{N_H}$ is the size of the hashtag set. We recommend n hashtags with the highest probability for post $\mathbf{p}_t$.

4 Experiment

4.1 Dataset

To perform the study, we collected data from 2,000 users (referred to as $\mathcal{U}$ in the following) on Twitter. For each user, we crawled his/her tweets published from 2015/1/1 to 2015/2/28 as training data and tweets published from 2015/3/1 to 2015/3/10 as testing data. This results in 127,848 training tweets and 17,276 testing tweets. Table 1 lists some statistical information about this dataset.

4.2 Compared Methods

We first compared the proposed model with several state-of-the-art methods, including methods that do not model the post history and those that model the short-term post history:

- **IBM1** [2]: IBM1 applies a translation model to obtain the alignment probability between the word and the tag.
- **TopicWA** [14]: TopicWA is a topical word alignment model, in which the standard LDA is employed to discover the latent topic.
- **Tweet2Vec** [8]: It applies a LSTM framework to the character sequence of a tweet to obtain its vector representations and predict a hashtag using the encoded vector.
- **LSTM-Attention** [19]: This is an attention-based LSTM model, which incorporates an LDA-based topic model into the LSTM architecture through an attention mechanism.
- **TPLDA** [10]: This is an LDA-based time-aware personalized hashtag recommendation model. It models the short-term post history.
- **HMemN2N** [11]: HMemN2N is a hierarchy end-to-end memory network based model. It constructs the post history with a fixed number (we adjusted this value on the developing data set) of latest historical posts and stores them in an end-to-end memory.

Then we explored the effectiveness of some components of the proposed model. To this end, we implemented the following variants of PLSTM.

- **PLSTM−Post History**: This variant does not model the post history for the hashtag recommendation. For every post $\mathbf{p}_t$, the variant performs the recommendation based on its content representation $\mathbf{x}_t$, with $\mathrm{score}(\mathbf{p}_t, \mathbf{H}_i) = \mathbf{x}_t^{\mathrm{T}}\mathbf{H}_i$.
- **PLSTM−Hashtag History**: This variant models the post history using the standard LSTM model. It does not separately models the hashtag history. Every hashtag within the post content is treated as a normal word with a randomly initialized embedding representation.

4.3 Implementation Details

We implemented the TopicWA, Tweet2Vec, TPLDA, and HMemN2N models with the code provided by their corresponding authors, and reimplemented other baselines. Hyper-parameters of these models (e.g., topic number for TopicWA, and memory size for HMemN2N) were adjusted on the developing data set. For the proposed model and its variants, the embedding dimension of the characters and words were set to 50, 300 respectively. We initialized the word embeddings with Google word2vec[1] [22]. For the post content encoding, we used 200 filters for each n-gram size $\in \{1, 2, 3, 4\}$. Hidden size of the recurrent network was set to 100. Dropout was applied to the word embeddings with a dropping probability of 0.5. For parameter updating, we used the Adadelta [23] optimizer with the default settings of Blocks[2].

[1] https://code.google.com/p/word2vec/

[2] http://blocks.readthedocs.io/en/latest/index.html

Table 2: Comparison of the proposed model with state-of-the-art methods and two variants of it on the test data set. Models with the † marker were implemented with the source code provided by their corresponding authors. The first four baselines did not model the post history, while the following two baselines modeled the short-term post history. The first variant (PLSTM−Post History) did not modeled the post history. The second variant (PLSTM−Hashtag History modeled) the whole post history but did not separately model the hashtags history.

Models	Hits@1	Hits@5
IBM1 [2]	0.2322	0.3043
TopicWA† [14]	0.3023	0.3975
Tweet2Vec† [8]	0.3116	0.4021
LSTM-Attention [19]	0.3413	0.4430
TPLDA† [10]	0.2737	0.5359
HMemN2N† [11]	0.3843	0.5460
PLSTM−Post History	0.3233	0.4350
PLSTM−Hashtag History	0.4151	0.6137
PLSTM (proposed)	**0.4671**	**0.6645**

4.4 Evaluation Metric

There are several evaluation metrics for hashtag recommendation, including Hits@N [6, 24], Precision, Recall, and F1 [6, 11]. Because in the setting of this work, there is only one ground truth hashtag for every recommendation, we choose the Hits@N metric to evaluate the model performance. The definition is precisely specified as follows:

$$\text{Hits@N} = \frac{\text{Number of Hits}}{\text{Recommendation times}}.$$

Here a hit occured when the recommended N hashtags include the ground truth hashtag.

4.5 Results and Discussion

Table 2 lists the results of the proposed model compared to those of the state-of-the-art baselines and its variants. From the table, we can obtain the following observations: (1) Our proposed model PLSTM consistently outperforms all of the state-of-the-art methods. This indicates the robustness and effectiveness of our approach. (2) For post content modeling, the neural network based models Tweet2Vec, LSTM-Attention, and PLSTM−Post History generally perform better than the LDA-based model TopicWA. This explains the popularity of neural network based approaches for this task. (3) Models modeling the post history (e.g., TPLDA, HMemN2N and PLSTM−Hashtag History) generally outperform those not modeling the post history, especially on Hits@5. This proves the informativeness of the post history for hashtag recommendation. (4) The long-term post history can bring additional improvement to the recommendation system. For example, compared to the HMemN2N model, there is an approximately 3% absolute improvement on Hits@1 and 5% absolute improvement on Hits@5 for our variant PLSTM−Hashtaq History, which models the long-term post history. This empirically verified our assumption that the long-term post history should be informative for

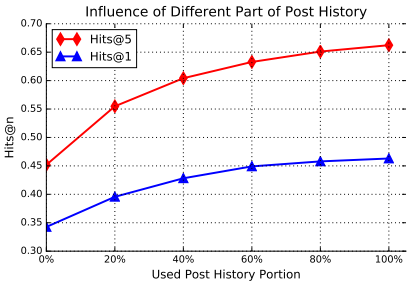

Fig. 2: Performance of the proposed model on test data set using different portions of post history. The 20% refers to the 20% of the historical posts closest in time to the testing data set, and it degenerates to the PLSTM-Post History model when the history portion is 0%.

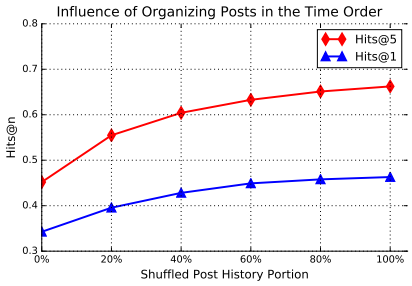

Fig. 3: Performance of the proposed model on test data set when shuffling different part of training data. The 20% refers to the 20% of the historical posts closest in time to the testing data set, and shuffling the data means not organizing the posts in the time order.

the current recommendation. (5) Additionally modeling the hashtag history can bring further improvement to the system. This is observed from the comparison between the proposed model and its variant PLSTM−Hashtag History.

4.6 Further Analysis

Influence of Different Part of Post History. From the above study, we know that the long-term post history can indeed improve the hashtag recommendation quality at the current time step. However, we do not know how much influence each part of the post history has on the final recommendation performance. To explore this question, we performed a study on different part of the post history. Specifically, during model inference, we removed the training data of different time steps and re-generated the memory content $\mathbf{c}_{\mathrm{N}_u}$ and $\mathbf{h}_{\mathrm{N}_u}$ using the trained model for each user with the left training data. Note that we did not re-train the model but only re-generated the memory content. Based on the resulting memory content, we tested our proposed model on the testing data set. Figure 2 shows the results of the proposed model using the latest 20%, 40%, 60%, 80% and 100% of the the post history for each user.

From the figure, we can see that the performance of our proposed model continuously increases as the post history increases. In addition, we can find that the improvement speed by the size of the post history slowly decreases as the time gap between the added historical posts and the test data set increases. For example, with the latest 20% of the post history, the performance increases by approximately 5% for Hits@1 and 10% for Hits@5. While additionally using the 20%-40% of the post history, the performance only increases approximately 3% for Hits@1 and 5% for Hits@5. We argue that this is because the latest 20% of the post history has a greater influence on the testing data than the 20%-40% of the post history. Similar observations could be obtained from comparisons between other portions. From these observations, we can see that the influence of the post history on the current recommendation decreases over

time. This also explains why it cannot greatly increase the memory size of HMemN2N, which treats every historical post equally.

Influence of Organizing Posts in Time Order. As previously mentioned, to make the recurrent neural network applicable, we organize the posts of a user in time order. A natural question is whether it is necessary to organize the historical posts in time order, or in other words, whether there is somehow dependency between contiguous posts. To answer this question, we designed the following experiment. For each user and his/her corresponding training data $\{\mathbf{p}_1, \mathbf{p}_2, \cdots, \mathbf{p}_{\mathrm{N}_u}\}$, we first shuffle the order of the latest 20%, 40%, 60%, 80% and 100% of the training data. Then, we re-trained the proposed model on this training data set and tested on the original testing data set. The results are shown in Figure 3.

From the figure, we can observe that shuffling the order of the training data degrades the testing performance. This indicates that there is indeed a dependency relation between contiguous posts. And similarly, we can also observe that it has a greater influence on the testing performance when the shuffled training data are closer in time to the testing data set. This verified again that the more recent post history is more informative for the recommendation. From these observations, we can come to the a conclusion that organizing historical posts in time order is indeed helpful to our system.

Inference Speed. Because this model was designed for real-time recommendation, it was critical for it to be time efficient. This section considers its time efficiency for inference. We supposed that recommendations had to be performed user by user and time by time. Thus, it was not possible to run the program in parallel. Therefore, we generated the content embedding $\mathbf{x}_t$ and performed recommendations sample by sample, instead of grouping them into a batch and considering them together. Table 3 lists the inference speeds on a GPU (NVIDIA TITAN X) and CPU (Intel(R) Xeon(R) CPU E5-2650 v3 @ 2.30GHz) with Theano [25]. As can be seen, even on a CPU, without much speed optimization, the average recommendation time is less than 0.4 s.

#Sample	Device	ms/Example
13,246	GPU	9.374
13,246	CPU	362.244

Table 3: Time efficiency of the proposed model for inference.

Initialization	Hits@1	Hits@5
Google	**0.4630**	**0.6623**
Random	0.4541	0.6567

Table 4: Performance of the proposed model PLSTM+FM with word embeddings initialized with Google word2vecs or randomly.

Parameter Sensitivity. In this section, we want to investigate the hyper-parameter influence on the performance of our proposed model. We first studied the influence of the pre-trained word embeddings. Table 4 lists the results of our proposed model with and without pre-trained word embeddings. As shown by the results, pre-trained word embeddings only provide a small benefit to the performance.

Another hyper-parameter of interest is the filter number N_f for each n-gram size. We tried different settings for $N_f \in \{100, 150, 200, 250\}$. The results listed in Table

5 show that the proposed model is non-sensitive to the variation of the filter number, especially on Hits@5. Considering the computation cost and performance, it is recommended to set $N_f \in (100, 250)$.

Table 5: Performance of the proposed model PLSTM+FM with different filter numbers.

FilterNum	Hits@1	Hits@5
100	0.4623	0.6648
150	0.4563	**0.6657**
200	**0.4630**	0.6623
250	0.4600	0.6618

5 Conclusion

The work aimed to provide a keyword-suggestion-like hashtag recommends, which recommends several hashtags when the user types in the hashtag (#) symbol. In contrast to previously published approaches, which did not consider the user's post history or only considered a few of the latest posts, the proposed model utilized the entire post history of the user to perform the recommendation. To this end, we organized the historical posts of the user in the time order and proposed a recurrent-neural-network-based model to perform the post history modeling. Experimental results on a dataset crawled from Twitter showed that the proposed model could achieve state-of-the-art performance.

References

1. Sedhai,S.,andSun,A. 2014. Hashtagrecommendation for hyperlinked tweets. In Proceedings of the 37th international ACM SIGIR conference on Research & development in information retrieval, 831–834. ACM.
2. Liu, Z.; Chen, X.; and Sun, M. 2011. A simple word trigger method for social tag suggestion. In Proceedings of the Conference on Empirical Methods in Natural Language Processing, 1577–1588. Association for Computational Linguistics.
3. Kowald, D.; Pujari, S. C.; and Lex, E. 2017. Temporal effects on hashtag reuse in twitter: A cognitive-inspired hashtag recommendation approach. In WWW, 1401– 1410.
4. Gong, Y.; Zhang, Q.; Han, X.; and Huang, X. 2017. Phrase-based hashtag recommendation for microblog posts. Science China Information Sciences 60(1):012109.
5. Dey, K., Shrivastava, R., Kaushik, S., Subramaniam, L. V.: Emtagger: a word embedding based novel method for hashtag recommendation on twitter. arXiv preprint arXiv:1712.01562. (2017)
6. She, J., and Chen, L. 2014. Tomoha: Topic model-based hashtag recommendation on twitter. In WWW, 371–372. ACM.
7. Zhao, F.; Zhu, Y.; Jin, H.; and Yang, L. T. 2016. A personalized hashtag recommendation approach using lda-based topic model in microblog environment. Future Generation Computer Systems 65:196–206.
8. Dhingra, B., Zhou, Z., Fitzpatrick, D., Muehl, M., Cohen, W. W. Tweet2vec: Character-based distributed representations for social media. arXiv preprint arXiv:1605.03481. (2016)

9. Gong, Y., and Zhang, Q. 2016a. Hashtag recommendation using attention-based convolutional neural network. In Proceedings of the 25th International Joint Conference on Artificial Intelligence (IJCAI 2016).
10. Zhang, Q.; Gong, Y.; Sun, X.; and Huang, X. 2014. Time-aware personalized hashtag recommendation on social media. In COLING, 203–212.
11. Huang, H.; Zhang, Q.; Gong, Y.; and Huang, X. 2016. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In Proceedings of COLING 2016: Technical Papers, 943–952. Osaka, Japan: The COLING 2016 Organizing Committee.
12. Heymann,P.;Ramage,D.;andGarcia-Molina,H. 2008. Social tag prediction. In SIGIR, 531–538. ACM.
13. Krestel, R.; Fankhauser, P.; and Nejdl, W. 2009. Latent dirichlet allocation for tag recommendation. In Proceedings of the third ACM conference on Recommender systems, 61–68. ACM.
14. Ding, Z., Zhang, Q., Huang, X: Automatic hashtag recommendation for microblogs using topicspecific translation model. In: 24th International Conference on Computational Linguistics, pp. 265. Citeseer. (2012)
15. Godin, F., Slavkovikj, V., De Neve, W., Schrauwen, B., Van de Walle, R.: Using topic models for twitter hashtag recommendation. In: Proceedings of the22ndInternationalConferenceonWorldWideWeb, pp. 593–596. ACM. (2013)
16. Ma, Z.; Sun, A.; Yuan, Q.; and Cong, G. 2014. Tagging your tweets: A probabilistic modeling of hashtag annotation in twitter. In Proceedings of the 23rd ACM International Conference on CIKM, 999– 1008. ACM.
17. Ding, Z., Qiu, X., Zhang, Q., and Huang, X.: Learning topical translation model for microblog hashtag suggestion. In: IJCAI. (2013)
18. Tomar,A.;Godin,F.;Vandersmissen,B.;DeNeve,W.; and Van de Walle, R. 2014. Towards twitter hashtag recommendation using distributed word representations and a deep feed forward neural network. In ICACCI, 362–368. IEEE.
19. Li, Y.; Liu, T.; Jiang, J.; and Zhang, L. 2016. Hashtag recommendation with topical attention-based lstm. In Proceedings of COLING 2016, 3019–3029.
20. Wang,Y.;Qu,J.;Liu,J.;Chen,J.;andHuang,Y. 2014. What to tag your microblog: Hashtag recommendation based on topic analysis and collaborative filtering. In Asia-Pacific Web Conference, 610–618. Springer.
21. Kim, Y. 2014. Convolutional neural networks for sentence classification. arXiv preprint arXiv:1408.5882.
22. Peng, X., and Gildea, D. 2016. Exploring phrasecompositionality in skip-gram models. arXiv preprint arXiv:1607.06208.
23. Zeiler, M. D. 2012. Adadelta: an adaptive learning rate method. arXiv preprint arXiv:1212.5701.
24. Kywe, S.; Hoang, T.-A.; Lim, E.-P.; and Zhu, F. 2012. On recommending hashtags in twitter networks. Social Informatics 337–350.
25. Theano Development Team. 2016. Theano: A Python framework for fast computation of mathematical expressions. arXiv e-prints abs/1605.02688.