Keyphrase Extraction Using Deep Recurrent Neural Networks on Twitter

Qi Zhang, Yang Wang, Yeyun Gong, Xuanjing Huang
Shanghai Key Laboratory of Data Science
School of Computer Science, Fudan University
Shanghai, P.R. China
{qz, ywang14, yygong12, xjhuang}@fudan.edu.cn

Abstract

Keyphrases can provide highly condensed and valuable information that allows users to quickly acquire the main ideas. The task of automatically extracting them have received considerable attention in recent decades. Different from previous studies, which are usually focused on automatically extracting keyphrases from documents or articles, in this study, we considered the problem of automatically extracting keyphrases from tweets. Because of the length limitations of Twitter-like sites, the performances of existing methods usually drop sharply. We proposed a novel deep recurrent neural network (RNN) model to combine keywords and context information to perform this problem. To evaluate the proposed method, we also constructed a large-scale dataset collected from Twitter. The experimental results showed that the proposed method performs significantly better than previous methods.

1 Introduction

Keyphrases are usually the selected phrases that can capture the main topics described in a given document (Turney, 2000). They can provide users with highly condensed and valuable information, and there are a wide variety of sources for keyphrases, including web pages, research articles, books, and even movies. In contrast to keywords, keyphrases usually contain two or more words. Normally, the meaning representations of these phrases are more precise than those of single words. Moreover, along with the increasing development of the internet, this kind of summarization has received continuous consideration in recent years from both the academic and entiprise communities (Witten et al., 1999; Wan and Xiao, 2008; Jiang et al., 2009; Zhao et al., 2011; Tuarob et al., 2015).

Because of the enormous usefulness of keyphrases, various studies have been conducted on the automatic extraction of keyphrases using different methods, including rich linguistic features (Barker and Cornacchia, 2000; Paukkeri et al., 2008), supervised classification-based methods (Witten et al., 1999; Wu et al., 2005; Wang et al., 2006), ranking-based methods (Jiang et al., 2009), and clustering-based methods (Mori et al., 2007; Danilevsky et al., 2014). These methods usually focus on extracting keyphrases from a single document or multiple documents. Typically, a large number of words exist in even a document of moderate length, where a few hundred words or more is common. Hence, statistical and linguistic features can be considered to determine the importance of phrases.

In addition to the previously mentioned methods, a few researchers have studied the problem of extracting keyphrases from collections of tweets (Zhao et al., 2011; Bellaachia and Al-Dhelaan, 2012). In contrast to traditional web applications, Twitter-like services usually limit the content length to 140 characters. In (Zhao et al., 2011), the context-sensitive topical PageRank method was proposed to extract keyphrases by topic from a collection of tweets. NE-Rank was also proposed to rank keywords for the purpose of extracting topical

Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, pages 836–845,
Austin, Texas, November 1-5, 2016.

keyphrases (Bellaachia and Al-Dhelaan, 2012). Because multiple tweets are usually organized by topic, many document-level approaches can also be adopted to achieve the task. In contrast with the previous methods, Marujo et al. (2015) focused on the task of extracting keywords from single tweets. They used several unsupervised methods and word embeddings to construct features. However, the proposed method worked on the word level.

In this study, we investigated the problem of automatically extracting keyphrases from single tweets. Compared to the problem of identifying keyphrases from documents containing hundreds of words, the problem of extracting keyphrases from a single short text is generally more difficult. Many linguistic and statistical features (e.g., the number of word occurrences) cannot be determined and used. Moreover, the standard steps of keyphrase extraction usually include keyword ranking, candidate keyphrase generation, and keyphrase ranking. Previous works usually used separate methods to handle these steps. Hence, the error of each step is propagated, which may highly impact the final performance. Another challenge of keyphrase extraction on Twitter is the lack of training and evaluation data. Manual labelling is a time-consuming procedure. The labelling consistency of different labellers cannot be easily controlled.

To meet these challenges, in this paper, we propose a novel deep recurrent neural network (RNN) model for the joint processing of the keyword ranking, keyphrase generation, and keyphrase ranking steps. The proposed RNN model contains two hidden layers. In the first hidden layer, we capture the keyword information. Then, in the second hidden layer, we extract the keyphrases based on the keyword information using a sequence labelling method. In order to train and evaluate the proposed method, we also proposed a novel method to construct a dataset that contained a large number of tweets with golden standard keyphrases. The proposed dataset construction method was based on the hashtag definitions in Twitter and how these were used in specific tweets.

The main contributions of this work can be summarized as follows:

- We proposed a two-hidden-layer RNN-based method to jointly model the keyword ranking, keyphrase generation, and keyphrase ranking steps.
- To train and evaluate the proposed method, we proposed a novel method for constructing a large dataset, which consisted of more than one million words.
- Experimental results demonstrated that the proposed method could achieve better results than the current state-of-the-art methods for these tasks.

2 Proposed Methods

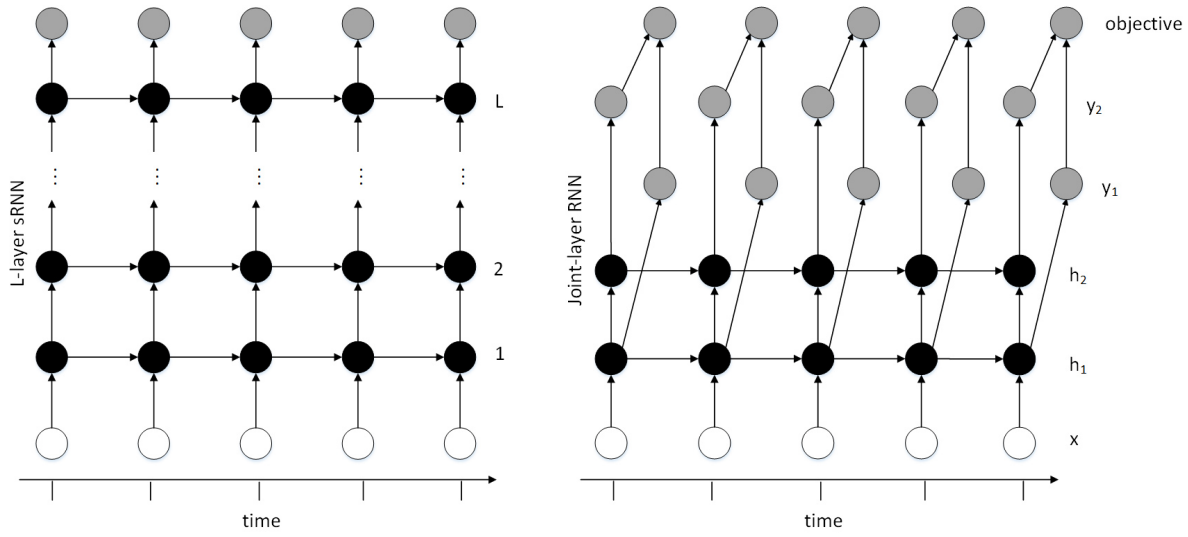
In this paper, we will first describe the deep recurrent neural network (RNN). Then, we will discuss the proposed joint-layer recurrent neural network model, which jointly processes the keyword ranking, keyphrase generation, and keyphrase ranking.

2.1 Deep Recurrent Neural Networks

One way to capture the contextual information of a word sequence is to concatenate neighboring features as input features for a deep neural network. However, the number of parameters rapidly increases according to the input dimension. Hence, the size of the concatenating window is limited. A recurrent neural network (RNN) can be considered to be a deep neural network (DNN) with an indefinite number of layers, which introduces the memory from previous time steps. A potential weakness of a RNN is its lack of hierarchical processing for the input at the current time step. To further provide hierarchical information through multiple time scales, deep recurrent neural networks (DRNNs) are explored (Hermans and Schrauwen, 2013). Fig. 1 (a) shows an L intermediate layer DRNN with full temporal connections (called a stacked RNN (sRNN) in (Pascanu et al., 2013)).

2.2 Joint-layer Recurrent Neural Networks

The proposed joint-layer recurrent neural network (joint-layer RNN) is a variant of an sRNN with two hidden layers. The joint-layer RNN has two output layers, which are combined into a objective layer. Suppose there is an L intermediate layer sRNN that has an output layer for each hidden layer. The l-th



(a) (b)

Figure 1: Deep recurrent neural network (DRNN) architectures: arrows represent connection matrices; white, black, and grey circles represent input frames, hidden states, and output frames, respectively; (a): L intermediate layer DRNN with recurrent connections at all levels (called stacked RNN); (b): joint-layer RNN folded out in time. Each hidden layer can be interpreted to be an RNN that receives the time series of the previous layer as input, where the hidden layer transforms into an output layer. Two output layers are combined via linear superposition into the objective function.

hidden activation is defined as:

$$\begin{aligned} \mathbf{h}_t^l &= f_h(\mathbf{h}_t^{l-1}, \mathbf{h}_{t-1}^l) \\ &= \phi_l(\mathbf{U}^l \mathbf{h}_{t-1}^l + \mathbf{W}^l \mathbf{h}_t^{l-1}), \end{aligned} \quad (1)$$

where $\mathbf{h}_t^l$ is the hidden state of the l-th layer at time t. $\mathbf{U}^l$ and $\mathbf{W}^l$ are the weight matrices for the hidden activation at time $t-1$ and the lower level activation $\mathbf{h}_t^{l-1}$, respectively. When $l = 1$, the hidden activation is computed using $\mathbf{h}_t^0 = \mathbf{x}_t$. ϕ_l is an element-wise non-linear function, such as the sigmoid function. The l-th output activation is defined as:

$$\begin{aligned} \hat{\mathbf{y}}_t^l &= f_o(\mathbf{h}_t^l) \\ &= \varphi_l(\mathbf{V}^l \mathbf{h}_t^l), \end{aligned} \quad (2)$$

where $\mathbf{V}^l$ is the weight matrix for the l-th hidden layer $\mathbf{h}_t^l$. φ_l is also an element-wise non-linear function, such as the softmax function.

A joint-layer recurrent neural network is an extension of a stacked RNN with two hidden layers. At time t, the training input, $\mathbf{x}_t$, of the network is the concatenation of features from a mixture within a window. We use word embedding as a feature in this paper. The output targets, $\mathbf{y}_t^1$ and $\mathbf{y}_t^2$, and output predictions, $\hat{\mathbf{y}}_t^1$ and $\hat{\mathbf{y}}_t^2$, of the network indicate whether the current word is a keyword and part of a keyphrase, respectively. $\hat{\mathbf{y}}_t^1$ just has two values $True$ and $False$ indicating whether the current word is keyword. $\hat{\mathbf{y}}_t^2$ has 5 values $Single$, $Begin$, $Middle$, End and Not indicating the current word is a single keyword, the beginning of a keyphrase, the middle (neither beginning nor ending) of a keyphrase, the ending of a keyphrase or not a part of a keyphrase.

Since our goal is to extract a keyphrase from a word sequence, we adopt a framework to simultaneously model keyword finding and keyphrase extraction. Figure 1 (b) shows the architecture of our model. The hidden layer formulation is defined as:

$$\mathbf{h}_t^1 = f_h(\mathbf{x}_t, \mathbf{h}_{t-1}^1) \quad (3)$$

$$\mathbf{h}_t^2 = f_h(\mathbf{h}_t^1, \mathbf{h}_{t-1}^2). \quad (4)$$

The output layer formulation is defined as:

$$\hat{\mathbf{y}}_t^1 = f_o(\mathbf{h}_t^1) \quad (5)$$

$$\hat{\mathbf{y}}_t^2 = f_o(\mathbf{h}_t^2). \quad (6)$$

2.3 Training

In this work, we joined learning the parameters θ in the deep neural network.

$$\boldsymbol{\theta} = \{\mathbf{X}, \mathbf{W}^1, \mathbf{W}^2, \mathbf{U}^1, \mathbf{U}^2, \mathbf{V}^1, \mathbf{V}^2\},$$

where $\mathbf{X}$ are the words embeddings, the other parameters are defined before. Once give a labeled sentence we can know both the keyword and keyphrase (keyphrase is made of keywords). At the first output layer we use our model to discriminate keyword and at the second output layer we use our model to discriminate keyphrase. Then we combine these two sub-objective which at different discrimination level into the final objective. The final objection is defined as:

$$J(\boldsymbol{\theta}) = \alpha J_1(\boldsymbol{\theta}) + (1-\alpha) J_2(\boldsymbol{\theta}), \tag{7}$$

where α is linear weighted factor. Given N training sequences $D = \left\{ \left(\mathbf{x}_t, \mathbf{y}_t^1, \mathbf{y}_t^2\right)_{t=1}^{T_n} \right\}_{n=1}^{N}$, the sub-objective formulation is defined as:

$$J_1(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} d(\hat{\mathbf{y}}_t^1, \mathbf{y}_t^1) \tag{8}$$

$$J_2(\boldsymbol{\theta}) = \frac{1}{N} \sum_{n=1}^{N} \sum_{t=1}^{T_n} d(\hat{\mathbf{y}}_t^2, \mathbf{y}_t^2), \tag{9}$$

where $d(\mathbf{a}, \mathbf{b})$ is a predefined divergence measure between **a** and **b**, such as Euclidean distance or cross-entropy.

Eq. (8) and Eq. (9) show that we discover keyword and extract keyphrase at different level simultaneously. The experimental results will show that combination of different granularity discrimination can significantly improve the performance.

To minimize the objective function, we optimize our models by back-propagating the gradients with respect to the training objectives. The stochastic gradient descent (SGD) algorithm is used to train the models. The update rule for the i-th parameter θ_i at epoch e is as follows:

$$\theta_{e,i} = \theta_{e-1,i} - \lambda g_{e,i}, \tag{10}$$

where the λ is a global learning rate shared by all dimensions. g_e is the gradient of the parameters at the e-th iteration. We select the best model according to the validation set.

#tweets	W	T	$\bar{N}_w$	$\bar{N}_t$
41,644,403	147,377	112,515	13.22	1.0

Table 1: Statistical information of dataset. W, T, $\bar{N}_w$, and $\bar{N}_t$ are the vocabulary of words, number of tweets with hashtags, average number of words in each tweet, and average number of hashtags in each tweet, respectively.

3 Experiments

3.1 Data Construction

To analyze the effectiveness of our model for keyphrase extraction on Twitter, we constructed an evaluation dataset. We crawled a large number of tweets. Generally, for each user, we gathered about 3K tweets, with a final total of more than 41 million tweets.

From analyzing these tweets, we found that some of the hashtags can be considered as the keyphrases of the tweet. For example: "The Warriors take Game 1 of the #NBAFinals 104-89 behind a playoff career-high 20 from Shaun Livingston.". "NBA Finals" can be considered as the keyphrase of the twitter. Based on this intuition, to construct the dataset, we firstly filtered out all non-Latin tweets using regular expressions. Then, we removed any URL links from the tweets since we were focusing on the textual content. Tweets that start with the "@username" are generally considered replies and have a conversational nature more than topical nature. Therefore, we also removed any tweets that start with "@username" to focus on topical tweets only. Moreover, we designed some rules about the hashtags in tweets to filter the remaining tweets. First, one tweet could have only one hashtag. Second, the position of the hashtag had to be inside the tweet because we needed the hashtag and tweet to be semantically inseparable. When a hashtag appears inside a tweet, it is most likely to be an inseparable semantical part of the tweet and has important meaning. Therefore, we regarded this hashtag as a keyphrase of the tweet.

Each hashtag was split into keywords if it encompassed more than one word, for example "OldStockCanadians" for "Old Stock Canadians". After an effort to filter the tweets we finally had 110K tweets with the hashtags which could meet our

Algorithm 1 Twitter Dataset Construction

Require: Tweets list $tList$
Ensure: Filtered Tweets and hashtags

```
resultList ← ∅
while t in tList do
    if t not contains latin letters then
        continue
    end if
    if t starts with "@username" then
        continue
    end if
    removed any URL links from t
    if t not exactly contains one hashtag then
        continue
    end if
    get hashtag from t
    split hashtag into keywords
    resultList.append((t, hashtag))
end while
return resultList
```

needs. The pseudocode is defined in Alg. 1. The statistical information of the dataset can be seen in Table 1. To evaluate the quality of the tweets in our dataset, we randomly selected 1000 tweets from our dataset and chose three volunteers. Every tweet was assigned a score of 2 (perfectly suitable), 1 (suitable), or 0 (unsuitable) to indicate whether the hashtag of the tweet was a good keyphrase for it. The results showed that 90.2% were suitable and 66.1% were perfectly suitable. This demonstrated that our constructed dataset was good for keyphrase extraction on Twitter.

3.2 Experiment Configurations

To perform an experiment on extracting keyphrases, we used 70% as a training set, 10% as a development set, and 20% as a testing set. For evaluation metrics, we used the precision (P), recall (R), and F1-score (F1) to evaluate the performance. The precision was calculated based on the percentage of keyphrases truly identified among the keyphrases labeled by the system. Recall was calculated based on the keyphrases truly identified among the golden standard keyphrases.

In the experiments, we use word embeddings as input to the neural network. The word embeddings we used in this work were pre-trained vectors trained on part of a Google News dataset (about 100 billion words). A skip-gram model (Mikolov et al., 2013) was used to generate these 300-dimensional vectors for 3 million words and phrases. We used the word embeddings to initialize our word weight matrix. The matrix was updated in the training process.

The default parameters of our model are as follows: The window size is 3, number of neurons in the hidden layer is 300, and α is 0.5, which were chosen based on the performance using the valid set.

3.3 Methods for Comparison

Several algorithms were implemented and used to evaluate the validity of the proposed approach. Among these algorithms, CRF, RNN, LSTM, and R-CRF treat the keyphrase extraction task as a sequence labelling task. Automatic keyword extraction on Twitter (AKET) uses an unsupervised method to extract keywords on Twitter.

- **CRF**: The keyphrase extraction task can be formalized as a sequence labeling task that involves the algorithmic assignment of a categorical label to each word of a tweet. CRF is a type of discriminative undirected probabilistic graphical model and can process a sequence labeling task. Hence, we applied CRF to extract keyphrases on Twitter.

- **RNN**: A recurrent neural network (RNN) is a type of artificial neural network where the connections between units form a directed cycle. This creates an internal state of the network that allows it to exhibit dynamic temporal behavior. In an RNN model, word embedding is introduced to represent the semantics of words.

- **LSTM**: Long short-term memory (LSTM) is a recurrent neural network (RNN) architecture. Unlike traditional RNNs, an LSTM network is well-suited to learn from experience to classify, process, and predict time series when there are very long time lags of unknown size between important events.

- **R-CRF**: A recurrent conditional random field (R-CRF)(Yao et al., 2014) is a mixture model

	P	R	F1
CRF	72.37%	71.82%	72.09%
RNN	78.65%	70.08%	74.14%
LSTM	77.52%	71.19%	74.22%
R-CRF	79.29%	73.15%	76.10%
AKET	11.00%	46.10%	17.80%
Joint-layer RNN	**80.74**%	**81.19**%	**80.97**%

Table 2: Keyphrase Extraction on Twitter

combining an RNN and a CRF. This model has the advantages of both the CRF and RNN. The previous work showed that the performance of R-CRF can be significantly improved.

- **AKET** (Automatic Keyword Extraction on Twitter) (Marujo et al., 2015): Several unsupervised methods and word embeddings were used to construct features to obtain keyword.

3.4 Experiment Results

Table 2 shows the performances of different methods on the dataset for keyphrase extraction. From the results, we observe that the joint-layer RNN achieved a better performance than the state-of-the-art methods. The relative improvement in the F-score of the joint-layer RNN over the second best result was 6.1%. AKET performed the worst. This was because AKET worked on the word level. Of the other methods, CRF performed the worst, RNN and LSTM were almost the same but better than CRF, and R-CRF was the best of these methods, with the exception of our joint-layer RNN. The results can be explained by the word embedding and long short-term memory cell providing some benefits. The best result was found with our joint-layer RNN. This indicated that the joint processing of the keyword finding and keyphrase extraction worked well and could to some degree demonstrate the effectiveness of our model in keyphrase extraction on Twitter.

To further analyze the keyword extraction results on Twitter, we compared AKET and our method. In Table 3, we can see that except for the recall, AKET is a little better than our method, but our method performed significantly better than AKET in the precision and F-score. This indicates that our model indeed has better performance in keyword finding.

	P	R	F1
AKET	20.68%	**87.56**%	33.46%
Joint-layer RNN	**87.45**%	85.38%	**86.40**%

Table 3: Keyword Extraction on Twitter

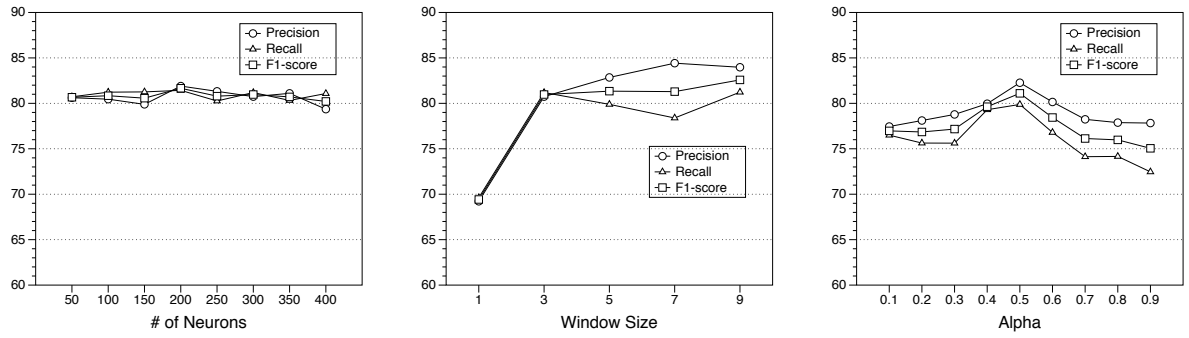
In summary, the experimental results conclusively demonstrated that the proposed joint-layer RNN method is superior to the state-of-the-art methods when measured using commonly accepted performance metrics on Twitter.

To analysis the sensitivity of the hyper-parameters of the joint-layer RNN, we conducted several empirical experiments on the dataset.

Fig.2(a) shows the performances of the joint-layer RNN with different numbers of neurons in the hidden layers. To simplify, we made hidden layer 1 and hidden layer 2 have the same number of neurons. In the figure, the x-axis denotes the number of neurons, and the y-axis denotes the precision, recall, and F-score. The data used for constructing the test set were the same as we used in the previous section. From the figure, we can observe that the number of neurons in the hidden layers do not highly affect the final performance. Three performance indicators of the joint-layer RNN change stably with different numbers of neurons.

Fig.2(b) shows the performances of the joint-layer RNN with different window sizes. In the figure, the x-axis denotes the different window size, and the y-axis denotes the precision, recall, and F-score. From the figure, we observe that when the window size is one, the three performance indicators of joint-layer RNN perform badly. Then, as the window size increases, the three performance indicators change stably. The main reason may possibly be that when the window size is one, the model just uses the current word information. When the window size increases, the model uses the context information of the current word but the most important context information is nearby the current word.

Fig.2(c) shows the performances of the joint-layer RNN with different α values. In the figure, the x-axis denotes the value of α used for training, and the y-axis denotes the precision, recall, and F-score.



(a) (b) (c)

Figure 2: (a): Performance with varying number of neurons in the hidden layer; (b): Performance with varying window size; (c): Performance with varying α.

	P	R	F1
WEU	80.74%	81.19%	80.97%
WENU	74.10%	69.30%	71.62%
REU	79.01%	79.75%	79.38%
RENU	78.16%	64.55%	70.70%

Table 4: Effects of embedding on performance. WEU, WENU, REU and RENU represent word embedding update, word embedding without update, random embedding update and random embedding without update respectively.

We can see that the best performance is obtained when α is around 0.5. This indicates that our model emphasizes the combination of keyword finding and keyphrase extraction.

Table 4 lists the effects of word embedding. We can see that the performance when updating the word embedding is better than when not updating, and the performance of word embedding is a little better than random word embedding. The main reason is that the vocabulary size is 147,377, but the number of words from tweets that exist in the word embedding trained on the Google News dataset is just 35,133. This means that 76.2% of the words are missing. This also confirms that the proposed joint-layer RNN is more suitable for keyphrase extraction on Twitter.

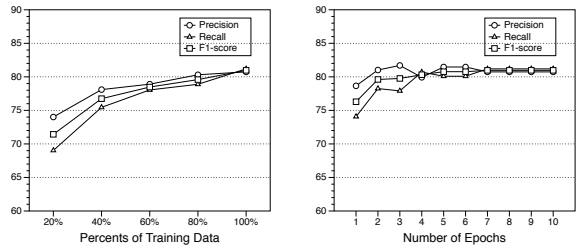
Fig.3(a) shows the performances of the joint-layer RNN with different percentages of training data. In the figure, the x-axis denotes the percentages of data used for training, and the y-axis denotes the precision, recall, and F-score. From the figure, we observe that as the amount of training data increases, the three performance indicators of the joint-layer RNN consequently improve. When the percentage of training data is greater than 60% of the whole dataset, the performance indicators slowly increase. The main reason may possibly be that the number concepts included in these data sets are small. However, on the other hand, we can say that the proposed joint-layer RNN method can achieve acceptable results with a few ground truths. Hence, it can be easily adopted for other data sets.

Since the keyphrase extraction training process is solved using an iterative procedure, we also evaluated its convergence property. Fig.3 (b) shows the precision, recall, and F-score performances of the joint-layer RNN. In the figure, the x-axis denotes the number of epochs for optimizing the model, and the y-axis denotes the precision, recall, and F-score. From the figure, we observe that the joint-layer RNN can coverage with less than six iterations. This means that the joint-layer RNN can achieve a stable and superior performance under a wide range of parameter values.

4 Related Work

In general, keyphrase extraction methods can be roughly divided into two groups: supervised machine learning approaches and unsupervised ranking approaches.

In the supervised line of research, keyphrase extraction is treated as a classification problem, in which a candidate must be classified as either a keyphrase (i.e., keyphrases) or not (i.e., non-



(a) (b)

Figure 3: (a): Effects of train size on performance; (b): Effects of the number of epochs on performance.

keyphrases). A classifier needs to be trained using annotated training data. The trained model is then applied to documents for which keyphrases are to be identified. For example (Frank et al., 1999) developed a system called KEA that used two features: tf-idf and first occurrence of the term and used them as input to Naive Bayes (Hulth, 2003) used linguistic knowledge (i.e., part-of-speech tags) to determine candidate sets: potential pos-patterns were used to identify candidate phrases from the text. Tang et al. (2004) applied Bayesian decision theory for keyword extraction. Medelyan and Witten extended the KEA to KEA++, which uses semantic information on terms and phrases extracted from a domain specific thesaurus, thus enhances automatic keyphrase extraction (Medelyan and Witten, 2006).

In the unsupervised line of research, keyphrase extraction is formulated as a ranking problem. A well-known approach is the Term Frequency Inverse Document Frequency (TF-IDF) (Sparck Jones, 1972; Zhang et al., 2007; Lee and Kim, 2008). Measures like term frequencies (Wu and Giles, 2013; Rennie and Jaakkola, 2005; Kireyev, 2009), inverse document frequencies, topic proportions, etc. and knowledge of specific domain are applied to rank terms in documents which are aggregated to score the phrases. The ranking based on tf-idf has been shown to work well in practice (Hasan and Ng, 2010). Mihalcea et al. proposed the TextRank, which constructs keyphrases using the PageRank values obtained on a graph based ranking model for graphs extracted from texts (Mihalcea and Tarau, 2004). Liu et al. proposed to extract keyphrases by adopting a clustering-based approach, which ensures that the document is semantically covered by these keyphrases (Liu et al., 2009). Ali Mehri et al. put forward a method for ranking the words in texts, which can also be used to classify the correlation range between word-type occurrences in a text, by using non-extensive statistical mechanics (Mehri and Darooneh, 2011).

Recurrent neural networks(RNNs) (Elman, 1990) has been applied to many sequential prediction tasks, which is an important class of naturally deep architecture. In NLP, RNNs deal with a sentence as a sequence of tokens and have been successfully applied to various tasks like spoken language understanding (Mesnil et al., 2013) and language modeling (Mikolov et al., 2011). Classical recurrent neural networks incorporate information from preceding, there are kinds of variants, bidirectional RNNs are also useful for NLP tasks, especially when making a decision on the current token, information provided by the following tokens is generally useful.

5 Conclusion

In this work, we proposed a novel deep recurrent neural network (RNN) model to combine keywords and context information to perform the keyphrase extraction task. The proposed model can jointly process the keyword ranking and keyphrase generation task. It has two hidden layers to discriminate keywords and classify keyphrases, and these two sub-objectives are combined into a final objective function. We evaluated the proposed method on a dataset filtered from ten million crawled tweets. The proposed method can achieve better results than the state-of-the-art methods. The experimental results demonstrated the effectiveness of the proposed method for keyphrase extraction on single tweets.

6 Acknowledgement

The authors wish to thank the anonymous reviewers for their helpful comments. This work was partially funded by National Natural Science Foundation of China (No. 61532011, 61473092, and 61472088), the National High Technology Research and Development Program of China (No. 2015AA015408).

References

Ken Barker and Nadia Cornacchia. 2000. Using noun phrase heads to extract document keyphrases. In *Advances in Artificial Intelligence*.

Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2012. Ne-rank: A novel graph-based keyphrase extraction in twitter. In *Proceedings of IEEE CS*.

Marina Danilevsky, Chi Wang, Nihit Desai, Xiang Ren, Jingyi Guo, and Jiawei Han. 2014. Automatic construction and ranking of topical keyphrases on collections of short documents. In *Proceedings of SDM*.

Jeffrey L Elman. 1990. Finding structure in time. *Cognitive science*.

Eibe Frank, Gordon W Paynter, Ian H Witten, Carl Gutwin, and Craig G Nevill-Manning. 1999. Domain-specific keyphrase extraction.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-art. In *Proceedings of COLING*.

Michiel Hermans and Benjamin Schrauwen. 2013. Training and analysing deep recurrent neural networks. In *Proceedings of NIPS*.

Anette Hulth. 2003. Improved automatic keyword extraction given more linguistic knowledge. In *Proceedings of EMNLP*.

Xin Jiang, Yunhua Hu, and Hang Li. 2009. A ranking approach to keyphrase extraction. In *Proceedings of SIGIR*.

Kirill Kireyev. 2009. Semantic-based estimation of term informativeness. In *Proceedings of NAACL*.

Sungjick Lee and Han-joon Kim. 2008. News keyword extraction for topic tracking. In *Proceedings of NCM*.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of EMNLP*.

Luis Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W Black, Anatole Gershman, David Martins de Matos, João Neto, and Jaime Carbonell. 2015. Automatic keyword extraction on twitter. In *Proceedings of ACL*.

Olena Medelyan and Ian H Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of JCDL*.

Ali Mehri and Amir H Darooneh. 2011. Keyword extraction by nonextensivity measure. *Physical Review E*.

Grégoire Mesnil, Xiaodong He, Li Deng, and Yoshua Bengio. 2013. Investigation of recurrent-neural-network architectures and learning methods for spoken language understanding. In *Proceedings of INTERSPEECH*.

Rada Mihalcea and Paul Tarau. 2004. Textrank: Bringing order into texts. In *Proceedings of ACL*.

Tomáš Mikolov, Stefan Kombrink, Lukáš Burget, Jan Honza Černockỳ, and Sanjeev Khudanpur. 2011. Extensions of recurrent neural network language model. In *Proceedings of ICASSP*.

Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Proceedings of NIPS*.

Junichiro Mori, Mitsuru Ishizuka, and Yutaka Matsuo. 2007. Extracting keyphrases to represent relations in social networks from web. In *Proceedings of IJCAI*.

Razvan Pascanu, Caglar Gulcehre, Kyunghyun Cho, and Yoshua Bengio. 2013. How to construct deep recurrent neural networks. *arXiv*.

Mari-Sanna Paukkeri, Ilari T Nieminen, Matti Pöllä, and Timo Honkela. 2008. A language-independent approach to keyphrase extraction and evaluation. In *Proceedings of COLING*.

Jason DM Rennie and Tommi Jaakkola. 2005. Using term informativeness for named entity detection. In *Proceedings of SIGIR*.

Karen Sparck Jones. 1972. A statistical interpretation of term specificity and its application in retrieval. *JDoc*.

Jie Tang, Juan-Zi Li, Ke-Hong Wang, and Yue-Ru Cai. 2004. Loss minimization based keyword distillation. In *Advanced Web Technologies and Applications*.

Suppawong Tuarob, Wanghuan Chu, Dong Chen, and Conrad S Tucker. 2015. Twittdict: Extracting social oriented keyphrase semantics from twitter. *IJCNLP*.

Peter D Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval*.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of AAAI*.

Jiabing Wang, Hong Peng, and Jing-song Hu. 2006. Automatic keyphrases extraction from document using neural network. In *Advances in Machine Learning and Cybernetics*.

Ian H Witten, Gordon W Paynter, Eibe Frank, Carl Gutwin, and Craig G Nevill-Manning. 1999. Kea: Practical automatic keyphrase extraction. In *Proceedings of DL*.

Zhaohui Wu and C Lee Giles. 2013. Measuring term informativeness in context. In *Proceedings of NAACL*.

Yi-fang Brook Wu, Quanzhi Li, Razvan Stefan Bot, and Xin Chen. 2005. Domain-specific keyphrase extraction. In *Proceedings of CIKM*.

Kaisheng Yao, Baolin Peng, Geoffrey Zweig, Dong Yu, Xiaolong Li, and Feng Gao. 2014. Recurrent conditional random field for language understanding. In *Proceedings of ICASSP*.

Yongzheng Zhang, Evangelos Milios, and Nur Zincir-Heywood. 2007. A comparative study on key phrase extraction methods in automatic web site summarization. *JDIM*.

Wayne Xin Zhao, Jing Jiang, Jing He, Yang Song, Palakorn Achananuparp, Ee-Peng Lim, and Xiaoming

Li. 2011. Topical keyphrase extraction from twitter.
In *Proceedings of ACL*.