Encoding Conversation Context for Neural Keyphrase Extraction from Microblog Posts

Yingyi Zhang[†*] **Jing Li**[‡] **Yan Song**[‡] **Chengzhi Zhang**[†]
[†]Nanjing University of Science and Technology
{yingyizhang, zhangcz}@njust.edu.cn
[‡]Tencent AI Lab
{ameliajli,clksong}@tencent.com

Abstract

Existing keyphrase extraction methods suffer from data sparsity problem when they are conducted on short and informal texts, especially microblog messages. Enriching context is one way to alleviate this problem. Considering that conversations are formed by reposting and replying messages, they provide useful clues for recognizing essential content in target posts and are therefore helpful for keyphrase identification. In this paper, we present a neural keyphrase extraction framework for microblog posts that takes their conversation context into account, where four types of neural encoders, namely, averaged embedding, RNN, attention, and memory networks, are proposed to represent the conversation context. Experimental results on Twitter and Weibo datasets[1] show that our framework with such encoders outperforms state-of-the-art approaches.

1 Introduction

The increasing popularity of microblogs results in a huge volume of daily-produced user-generated data. As a result, such explosive growth of data far outpaces human beings' reading and understanding capacity. Techniques that can automatically identify critical excerpts from microblog posts are therefore in growing demand. Keyphrase extraction is one of the techniques that can meet this demand, because it is defined to identify salient phrases, generally formed by one or multiple words, for representing key focus and main topics for a given collection (Turney, 2000; Zhao et al., 2011). Particularly for microblogs, keyphrase extraction has been proven useful to downstream applications such as information retrieval (Choi et al., 2012), text summarization (Zhao et al., 2011), event tracking (Ribeiro et al., 2017), etc.

Target post for keyphrase extraction:
"I will curse you in that forum" is the lowest of low. You are an embarrassment **president Duterte**. Childish!
Messages forming a conversation:
[R1]: any *head of state* will be irked if asked to report to *another head of state*
[R2]: Really? Did *Obama* really asked *Duterte* to report to him? LOL

Table 1: An example conversation about "president Duterte" on Twitter. [Ri]: The i-th message in conversation ordered by their positing time. **president Duterte**: keyphrase to be detected; *Italic words*: words that are related to the main topic in conversations and can indicate the keyphrase.

To date, most efforts on keyphrase extraction on microblogs treat messages as independent documents or sentences, and then apply ranking-based models (Zhao et al., 2011; Bellaachia and Al-Dhelaan, 2012; Marujo et al., 2015) or sequence tagging models (Zhang et al., 2016) on them. It is arguable that these methods are suboptimal for recognizing salient content from short and informal messages due to the severe data sparsity problem. Considering that microblogs allow users to form conversations on issues of interests by *reposting with comments*[2] and *replying to messages* for voicing opinions on previous discussed points, these conversations can enrich context for short messages (Chang et al., 2013; Li et al., 2015), and have been proven useful for identifying topic-related content (Li et al., 2016). For example, Table 1 displays a target post with keyphrase "*president Duterte*" and its reposting and replying messages forming a conversation.

Easily identified, critical words are mentioned multiple times in conversations. Such as in [R2], keyword "*Duterte*" re-occurs in the conversation.

[*]Work was done during the internship at Tencent AI Lab.
[1]Our datasets are released at: http://ai.tencent.com/ailab/Encoding_Conversation_Context_for_Neural_Keyphrase_Extraction_from_Microblog_Posts.html

[2]On Twitter, reposting behavior is named as retweet.

Also, topic-relevant content, e.g., "*head of state*", "*another head of state*", "*Obama*", helps to indicate keyphrase "*president Duterte*". Such contextual information embedded in a conversation is nonetheless ignored for keyphrase extraction in existing approaches.

In this paper, we present a neural keyphrase extraction framework that exploits conversation context, which is represented by neural encoders for capturing salient content to help in indicating keyphrases in target posts. Conversation context has been proven useful in many NLP tasks on social media, such as sentiment analysis (Ren et al., 2016), summarization (Chang et al., 2013; Li et al., 2015), and sarcasm detection (Ghosh et al., 2017). We use four context encoders in our model, namely, averaged embedding, RNN (Pearlmutter, 1989), attention (Bahdanau et al., 2014), and memory networks (Weston et al., 2015), which are proven useful in text representation (Cho et al., 2014; Weston et al., 2015; Huang et al., 2016; Nie et al., 2017). Particularly in this task, to the best of our knowledge, we are the first to encode conversations for detecting keyphrases in microblog posts. Experimental results on Twitter and Sina Weibo datasets demonstrate that, by effectively encoding context in conversations, our proposed approach outperforms existing approaches by a large margin. Quantitative and qualitative analysis suggest that our framework performs robustly on keyphrases with various length. Some encoders such as memory networks can detect salient and topic-related content, whose occurrences are highly indicative of keyphrases. In addition, we test ranking-based models with and without considering conversations. The results also confirm that conversation context can boost keyphrase extraction of ranking-based models.

2 Keyphrase Extraction with Conversation Context Encoding

Our keyphrase extraction framework consists of two parts, i.e., a keyphrase tagger and a conversation context encoder. The keyphrase tagger aims to identify keyphrases from a target post, and the context encoders captures the salient content in conversations, which would indicate keyphrases in the target post. The entire framework is learned synchronously with the given target posts and their corresponding conversation context. In prediction, the keyphrase tagger identifies keyphrases in a post with the help of representations generated by the encoder. Figure 1 shows the overall structure of our keyphrase extraction framework. In the rest of this section, Section 2.1 describes the keyphrase taggers used in our framework; Section 2.2 gives the details of different context encoders.

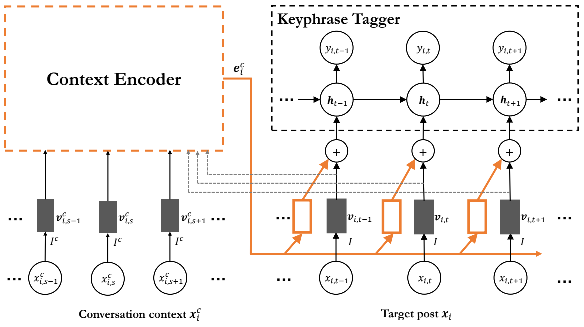


Figure 1: The overall structure of our keyphrase extraction framework with context encoder. Grey dotted array refer to the inputs of target posts that are also used in context encoding.

SINGLE	$x_{i,t}$ is a one-word keyphrase (keyword).
BEGIN	$x_{i,t}$ is the first word of a keyphrase.
MIDDLE	$x_{i,t}$ is part of a keyphrase but it is neither the first nor the last word of the keyphrase.
END	$x_{i,t}$ is the last word of a keyphrase
NOT	$x_{i,t}$ is not a keyword or part of a keyphrase.

Table 2: Definitions of different $y_{i,t}$.

2.1 Keyphrase Taggers

We follow Zhang et al. (2016) to cast keyphrase extraction into the sequence tagging task. Formally, given a target microblog post $\mathbf{x}_i$ formulated as word sequence $< x_{i,1}, x_{i,2}, ..., x_{i,|\mathbf{x}_i|} >$, where $|\mathbf{x}_i|$ denotes the length of $\mathbf{x}_i$, we aim to produce a tag sequence $< y_{i,1}, y_{i,2}, ..., y_{i,|\mathbf{x}_i|} >$, where $y_{i,t}$ indicates whether $x_{i,t}$ is part of a keyphrase. In detail, $y_{i,t}$ has five possible values:

$$y_{i,t} \in \{\text{SINGLE}, \text{BEGIN}, \text{MIDDLE}, \text{END}, \text{NOT}\}$$

Table 2 lists the definition of each value. Zhang et al. (2016) has shown that keyphrase extraction methods with this 5-value tagset perform better than those with binary outputs, i.e., only marked with yes or no for a word to be part of a keyphrase.

To predict keyphrase tags, we use four state-of-the-art neural sequence taggers, namely, recurrent neural networks (RNN) (Pearlmutter, 1989), RNN with gated recurrent units (GRU) (Chung et al., 2014), long short-term memory (LSTM) networks (Hochreiter and Schmidhuber, 1997), and bidirectional LSTM (BiLSTM) (Graves and Schmidhuber, 2005).

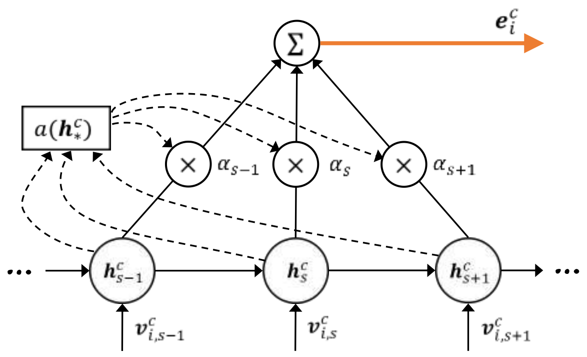


Figure 2: The structure of attention-based conversation context encoder.

In addition to one-type output, we also use joint-layer RNN proposed by Zhang et al. (2016), which is demonstrated to be the state-of-the-art keyphrase tagger in previous work without modeling conversation context. As a multi-task learner (Collobert and Weston, 2008), joint-layer RNN tackles two tasks with two types of outputs, $y^1_{i,t}$ and $y^2_{i,t}$. $y^1_{i,t}$ has a binary tagset, which indicates whether word $x_{i,t}$ is part of a keyphrase or not. $y^2_{i,t}$ employs the 5-value tagset defined in Table 2. Besides the standard RNN version, in implementation, we also build the joint-layer RNN with its GRU, LSTM, and BiLSTM counterparts. To be consistent, taggers with one-type output with the 5-value tagset are named as single-layer taggers.

As shown in Figure 1, our keyphrase tagger is built upon input feature map $I(\cdot)$, which embeds each word $x_{i,t}$ in target post into a dense vector format, i.e., $I(x_{i,t}) = \boldsymbol{\nu}_{i,t}$. We initialize input feature map by pre-trained embeddings, and update embeddings during training.

2.2 Context Encoders

We aggregate all reposting and replying messages in conversations to form a pseudo-document as *context* by their posting time, and input context in forms of word sequences into context encoder. Let $\mathbf{x}^c_i$ denote the context word sequence of the target post $\mathbf{x}_i$, we propose four methods to encode $\mathbf{x}^c_i$, namely, *averaged embedding*, *RNN*, *attention*, and *memory networks*. Similar to keyphrase taggers (see Section 2.1), each word $x^c_{i,s}$ in context $\mathbf{x}^c_i$ takes the form of a vector $\boldsymbol{\nu}^c_{i,s}$ mapped by an input layer $I^c(\cdot)$, which is also initialized by pre-trained embeddings, and updated in the training process.

2.2.1 Averaged Embedding

As a straightforward sentence representation technique, averaged embedding simply takes the average embeddings of words in a context, i.e., $\boldsymbol{\nu}^c_{i,s}$, as the encoding of context representation, i.e.,

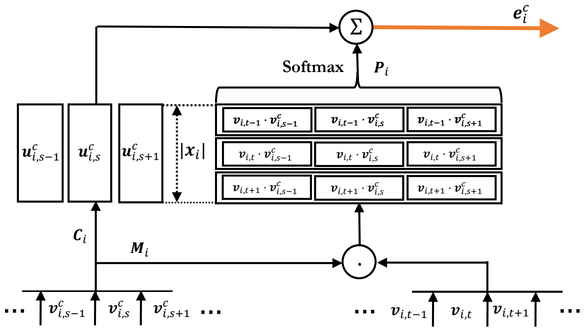


Figure 3: The structure of the conversation context encoder based on memory networks.

$$\mathbf{e}^c_i = \frac{1}{|\mathbf{x}^c_i|} \sum_{s=1}^{|\mathbf{x}^c_i|} \boldsymbol{\nu}^c_{i,s} \tag{1}$$

where $|\mathbf{x}^c_i|$ is the length of $\mathbf{x}^c_i$ in the context.

2.2.2 RNN

RNN encoders employ the recurrent neural network model for the embedded context sequence $< \boldsymbol{\nu}^c_{i,1}, \boldsymbol{\nu}^c_{i,2}, ..., \boldsymbol{\nu}^c_{i,|\mathbf{x}^c_i|} >$, through the recurrent functions over all the states:

$$\mathbf{h}^c_{i,s} = \delta_h(\mathbf{W}^1_h \mathbf{h}^c_{i,s-1} + \mathbf{W}^2_h \boldsymbol{\nu}^c_{i,s}) \tag{2}$$

where $\mathbf{W}^1_h$ and $\mathbf{W}^2_h$ are learnable weight matrices, and δ_h is the component-wise sigmoid function. The encoder representation is thus given by the hidden units at the last state:

$$\mathbf{e}^c_i = \mathbf{h}^c_{|\mathbf{x}^c_i|} \tag{3}$$

In this paper, RNN-based encoders have four variants, namely, RNN, GRU, LSTM, and BiLSTM. Particularly, as BiLSTM has two opposite directions, its context representation takes the concatenation of the last states from both directions, which come from two ends of a given context.

2.2.3 Attention

Attention-based encoders put attention mechanism (Bahdanau et al., 2014) upon RNN model for "soft-addressing" important words in the conversation context. In this paper, we use the feed-forward attention (Raffel and Ellis, 2015; SØnderby et al., 2015), as shown in Figure 2. The encoder is thus represented as

$$\mathbf{e}^c_i = \sum_{s=1}^{|x^c_i|} \alpha^c_{i,s} \mathbf{h}^c_{i,s} \tag{4}$$

where $\alpha^c_{i,s}$ is the attention coefficient obtained for word $\mathbf{x}^c_s$, which implicitly reflects its importance for helping keyphrase identification. $\alpha^c_{i,s}$ is computed via a softmax over the hidden states by

$$\alpha^c_{i,s} = softmax(a(\mathbf{h}^c_{i,s})) \tag{5}$$

where $a(\cdot)$ is a learnable function formulated as:

$$a(\mathbf{h}^c_{i,s}) = tanh(\mathbf{W}_a \mathbf{h}^c_{i,s}) \tag{6}$$

which takes input only from on $\mathbf{h}^c_{i,s}$. $\mathbf{W}_a$ are parameters of the function $a(\cdot)$ to be learned.

2.2.4 Memory Networks

The encoder based on memory networks (MemNN) (Weston et al., 2015) stores and updates the representations of conversation contexts in a memory module. The updated representations are used to guide the keyphrase tagger. Figure 3 illustrates its structure.

Formally, each embedded context sequence $\mathbf{V}^c_i =< \boldsymbol{\nu}^c_{i,1}, \boldsymbol{\nu}^c_{i,2}, ..., \boldsymbol{\nu}^c_{i,|\mathbf{x}^c_i|} >$ is stored into memory $\mathbf{M}_i$. We then yield the match between embedded target post $\mathbf{V}_i =< \boldsymbol{\nu}_{i,1}, \boldsymbol{\nu}_{i,2}, ..., \boldsymbol{\nu}_{i,|\mathbf{x}_i|} >$ and context memory $\mathbf{M}_i$ by their inner product activated by softmax:

$$\mathbf{P}_i = softmax(\mathbf{V}_i \cdot \mathbf{M}_i) \tag{7}$$

where $\mathbf{P}_{i,j,j'}$ captures the similarity between the j-th word in conversation context $\mathbf{x}^c_i$ and the j'-th word in target post $\mathbf{x}_i$.

To transform context input $\mathbf{x}^c_i$ into an aligned form so that it is able to be added with $\mathbf{P}_i$, we include another embedding matrix $\mathbf{C}_i =< \boldsymbol{\mu}_{i,1}, ..., \boldsymbol{\mu}_{i,|\mathbf{x}^c_i|} >$. Similar to attention encoder, the MemNN encoder aims to generate a representation, which addresses the important part in the conversation context that helps tagging keyphrases in target post $\mathbf{x}_i$. The sum of $\mathbf{C}_i$ and matching matrix $\mathbf{P}_i$ serves as the encoded representation for conversation context:

$$\mathbf{e}^c_i = \mathbf{P}_i + \mathbf{C}_i \tag{8}$$

In particular, both attention and MemNN explores salient words in conversations that describe main focus of the conversation, which helps indicate keyphrases of a target post. In comparison, MemNN explicitly exploits the affinity of target posts and conversations in matching each other, while attention implicitly highlights certain context without taking target posts into account.

Dataset	# of annot. msgs	# of msgs in context	Context length	Vocab
Twitter				
Train	3,976	3.38	49.74	34,412
Dev	497	3.19	46.44	7,186
Test	497	3.30	48.09	8,779
Weibo				
Train	13,816	1.97	55.77	25,259
Dev	1,727	2.01	45.00	9,106
Test	1,727	1.82	51.95	9,305

Table 3: Statistics of two datasets. Train, Dev, and Test denotes training, development, and test set, respectively. # of annot. msgs: number of messages with keyphrase annotation, each containing conversation context. # of msgs in context: average count of message in conversation context. Context length: average count of words in conversation context. Vocab: vocabulary size.

3 Experiment Setup

3.1 Datasets

Our experiments are conducted on two datasets collected from Twitter and Weibo[3], respectively. The **Twitter** dataset is constructed based on TREC2011 microblog track[4]. To recover conversations, we used Tweet Search API[5] to retrieve full information of a tweet with its "in reply to status id" included. Recursively, we searched the "in reply to" tweet till the entire conversation is recovered. Note that we do not consider retweet relations, i.e., reposting behaviors on Twitter, because retweets provide limited extra textual information for the reason that Twitter did not allow users to add comments in retweets until 2015. To build the **Weibo** dataset, we tracked real-time trending hashtags[6] on Weibo and used the hashtag-search API[7] to crawl the posts matching the given hashtag queries. In the end, a large-scale Weibo corpus is built containing Weibo messages posted during January 2nd to July 31st, 2014.

For keyphrase annotation, we follow Zhang et al. (2016) to use microblog hashtags as gold-

[3]Weibo is short for Sina Weibo, the biggest microblog platform in China and shares the similar market penetration as Twitter (Rapoza, 2011). Similar to Twitter, it has a length limitation of 140 Chinese characters.

[4]`http://trec.nist.gov/data/tweets/`

[5]`http://developer.twitter.com/en/docs/tweets/search/api-reference/get-saved_searches-show-id`

[6]`http://open.weibo.com/wiki/Trends/hourly`

[7]`http://www.open.weibo.com/wiki/2/search/topics`

	Single-layer Taggers				Joint-layer Taggers			
	RNN	GRU	LSTM	BiLSTM	RNN	GRU	LSTM	BiLSTM
No Encoder	44.9±1.4	53.9±4.7	54.9±3.8	60.8±3.6	51.0±3.3	56.1±3.4	55.1±2.6	62.5±0.9
Context Encoder								
Avg Emb	50.4±0.9	58.8±2.9	56.0±0.7	62.2±3.0	51.5±1.7	59.0±3.5	58.7±3.7	64.5±0.4
RNN	46.4±1.6	56.4±1.9	55.6±2.5	59.0±2.4	52.2±2.8	54.4±2.8	58.3±1.8	63.7±1.3
GRU	50.3±0.8	53.7±1.0	58.0±0.9	56.8±2.3	50.8±4.8	52.3±3.8	57.0±2.1	63.0±1.3
LSTM	51.6±2.0	56.4±1.4	57.9±2.3	**64.0**±3.1	50.8±3.1	57.9±2.3	58.3±4.0	64.2±0.6
BiLSTM	49.2±1.7	58.3±1.1	56.0±2.0	62.6±3.2	52.7±3.4	56.8±1.0	56.5±3.6	63.7±2.3
Att (LSTM)	48.7±1.7	58.1±1.7	58.1±3.1	**64.0**±1.8	51.7±4.8	57.4±2.3	58.0±2.4	63.8±1.5
Att (BiLSTM)	51.7±1.4	58.3±1.5	57.0±3.6	62.8±2.5	52.3±4.3	58.0±1.8	59.0±3.9	64.2±3.4
MemNN	**53.6**±0.3	**59.4**±3.1	**59.5**±4.1	62.4±4.8	**53.7**±3.5	**59.4**±2.1	**62.3**±3.3	**65.5**±1.6

Table 4: Comparisons of the average F1 scores (%) and their standard deviations measured on Twitter over the results of models with 5 sets of parameters for random initialization. The left half reports results of single-layer taggers; The right half reports results of joint-layer taggers. Each column: results of the same tagger with different encoders. Each row: results of different taggers with the same encoder. No Encoder: taggers without encoding context. Abbreviations for context encoders: Avg Emb – averaged embedding; Att (LSTM) – attention on LSTM; Att (BiLSTM) – attention on BiLSTM; MemNN – memory networks.

standard keyphrases[8] and filtered all microblog posts by two rules: first, there is only one hashtag per post; second, the hashtag is inside a post, i.e., containing neither the first nor the last word of a post. Then, we removed all the "#" symbols in hashtags before keyphrase extraction. For both Twitter and Weibo dataset, we randomly sample 80% for training, 10% for development, and the rest 10% for test. Table 3 reports the statistics of the two datasets. The dataset released by Zhang et al. (2016) is not used because it does not contain conversation information.

We preprocessed Twitter dataset with Twitter NLP tool[9] (Gimpel et al., 2011; Owoputi et al., 2013) for tokenization. For Weibo dataset, we used NLPIR tool[10] (Zhang et al., 2003) for Chinese word segmentation. In particular, Weibo conversations have an relatively wide range (from 3 to 8,846 words), e.g., one conversation could contain up to 447 messages. If use the maximum length of all conversations as the input length for encoders, padding the inputs will lead to a sparse matrix. Therefore, for long conversations (with more than 10 messages), we use KLSum (Haghighi and Vanderwende, 2009) to produce summaries with a length of 10 messages and then encode the produced summaries. In contrast, we do not summarize Twitter conversations because their length range is much narrower (from 4 to 1,035 words).

[8]Zhang et al. (2016) proves that 90% of the hashtag-annotated keyphrases match human annotations.

[9]`http://www.cs.cmu.edu/~ark/TweetNLP/`

[10]`https://github.com/NLPIR-team/NLPIR`

3.2 Model Settings

For keyphrase taggers based on RNN, GRU, and LSTM, we follow Zhang et al. (2016) and set their state size to 300. For the BiLSTM tagger, which has two directions, we set the state size for each direction to 150. The joint-layer taggers employ the same hyper-parameters according to Zhang et al. (2016). The state size of context encoders shares the same settings with keyphrase taggers. In training, the entire keyphrase extraction framework uses cross-entropy loss and RMSprop optimizer (Graves, 2013) for parameter updating.

We initialize input feature map I for target post and I_c for conversation context by embeddings pre-trained on large-scale external microblog collections from Twitter and Weibo. Twitter embeddings are trained on 99M tweets with 27B tokens and 4.6M words in the vocabulary. Weibo embeddings are trained on 467M Weibo messages with 1.7B words and 2.5M words in the vocabulary.

In comparison, we employ neural taggers without encoding conversation context, which are based on RNN, GRU, LSTM, and BiLSTM. We also compare our models with the state-of-the-art joint-layer RNN (Zhang et al., 2016) and its GRU, LSTM, and BiLSTM variations.

To further illustrate the effectiveness of leveraging conversation context for keyphrase extraction, we also evaluate some ranking-based models, namely, TF-IDF (Salton and Buckley, 1988), TextRank (Mihalcea and Tarau, 2004), and KEA implemented by KEA-3.0[11] (Witten et al., 1999).

[11]`www.nzdl.org/Kea/Download/KEA-3.0.zip`

	Single-layer Taggers				Joint-layer Taggers			
	RNN	GRU	LSTM	BiLSTM	RNN	GRU	LSTM	BiLSTM
No encoder	58.8±1.4	66.2±0.8	67.3±1.6	74.8±0.7	55.5±0.5	64.1±0.7	64.9±0.6	76.8±0.5
Context Encoder								
Avg Emb	**63.3**±0.9	68.2±0.7	69.4±0.4	76.6±0.9	61.1±1.2	69.7±1.3	69.3±0.7	79.8±0.6
RNN	58.2±1.6	64.9±0.6	65.3±0.8	73.1±0.1	60.9±0.5	67.1±0.6	66.7±0.5	71.2±0.7
GRU	56.5±0.8	67.0±0.6	67.4±1.1	73.8±0.7	58.4±1.1	65.5±0.8	67.1±0.4	76.2±0.7
LSTM	59.4±2.0	67.6±0.8	68.1±0.5	75.5±0.2	61.1±1.9	68.4±1.1	69.5±0.7	78.1±1.0
BiLSTM	60.8±1.7	68.6±1.0	68.4±0.7	75.9±0.7	61.6±1.8	69.3±1.0	69.6±0.3	78.2±0.8
Att (LSTM)	62.4±1.8	67.6±1.1	69.0±0.7	75.8±1.2	**63.1**±1.3	70.2±0.8	70.8±1.3	79.3±0.5
Att (BiLSTM)	59.6±1.4	68.6±0.6	**70.4**±1.0	76.5±0.8	61.5±2.2	**70.5**±0.6	**71.0**±0.5	**80.5**±1.7
MemNN	61.1±0.4	**69.3**±0.5	69.9±0.7	**79.1**±1.1	61.8±1.4	68.7±0.9	69.3±0.4	79.6±1.4

Table 5: Comparisons of F1 scores on Weibo. The abbreviations are defined the same as those in Table 4.

We design two experiment settings when running these models: 1) each target post is treated as a document; 2) each conversation (containing the target post) is treated as a document. We select the top N words for each target post by their ranked-orders and the threshold N is tuned on the development set. As a result, N ranges from 2 to 7 for various methods. Particularly, since TF-IDF and TextRank extract keywords instead of keyphrases, we aggregate the selected keywords according to Bellaachia and Al-Dhelaan (2012).

4 Experimental Results

Section 4.1 to 4.5 present quantitative and qualitative analysis of our neural keyprhase extraction models. Section 4.6 reports the performance of ranking-based models where we test the general applicability of incorporating conversation context to non-neural keyphase extraction methods.

4.1 Overall Comparisons

Table 4 and Table 5 report F1 scores on Twitter and Weibo, respectively.[12] We have the following observations.

Conversation context is useful for keyphrase extraction. By combining the encoded context in conversations, the F1 scores of all taggers are better than their basic versions without context encoders. It confirms that content in conversations helps in indicating keyphrases in target posts.

Selecting the correct context encoder is important. Encoding context simply by RNN or GRU yields poor results. The reason for RNN is that it suffers from gradient vanishing problem when encoding long conversations (conversions in our two datasets have over 45 words on average). The reason for GRU is that its forget gates may be not well trained to process important content when the training set is small.

The results of AvgEmb are the worst on Twitter while competitive to other encoders on Weibo. The performance of AvgEmb is competitive to other complex context encoders on Weibo. The reason may be that incorrect word orders generally do not affect the understanding in Chinese, where word order misuse is prevalent in Chinese Weibo messages. As a result, encoding word orders, as is done by the encoders except AvgEmb, might bring noise to keyphrase extraction on Weibo dataset. In contrast, AvgEmb is the worst encoder on Twitter dataset, as word order is crucial in English.

Identifying salient content in context is important. Four types of context encoders have different behaviors. Avg Emb considers all words in conversation context are equally important. RNN-variant context encoders, i.e., RNN, GRU, LSTM, and BiLSTM, additionally explore the relations between succeeded words without distinguishing salient and non-salient words. Attention (Att (LSTM) and Att (BiLSTM)) and MemNN can recognize critical content in conversations, which would indicate keyphrases in target posts. Therefore, our keyphrase extraction framework with attention or MemNN encoder has generally better F1 scores than those with other encoders.

MemNN can effectively capture salient content in context. On Twitter dataset, MemNN achieves the best F1 scores when combining with various keyphrase taggers except for single-layer GRU and BiLSTM. On Weibo dataset, although MemNN does not always outperform other encoders, its performance is close to the best ones.

[12]We also tried BiRNN and BiGRU as keyphrase taggers and as context encoders. They are outperformed by BiLSTM. We don't report these results due to the space limitation.

	SL BiLSTM		JL BiLSTM	
	Twitter	Weibo	Twitter	Weibo
No encoder	60.8	74.8	62.5	76.8
Avg Emb	61.0	75.7	63.3	79.2
RNN	58.8	72.7	63.1	71.1
GRU	56.4	73.1	62.7	76.0
LSTM	61.3	75.2	63.9	77.8
BiLSTM	62.0	75.4	62.3	78.0
Att (LSTM)	**62.6**	75.6	63.7	79.2
Att (BiLSTM)	62.0	76.5	63.9	**79.9**
MemNN	61.6	**77.4**	**65.1**	79.2

Table 6: The F1 scores of BiLSTM taggers measured on test instances without conversation context (%). SL BiLSTM and JL BiLSTM denote keyphrase tagger as single-layer and joint-layer BiLSTM, respectively. The other abbreviations are defined the same as those in Table 4.

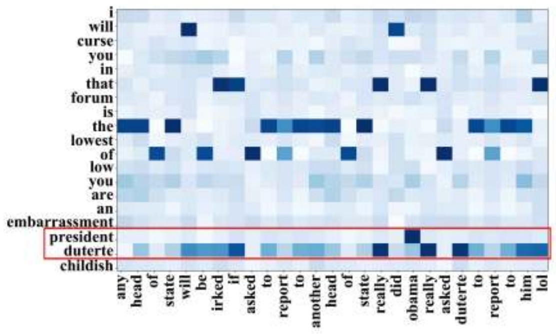


Figure 4: The heatmap of the context representation generated by MemNN (see Eq. 8). The horizontal axis refers to words in the conversation context, while the vertical axis refers to words in the target post. Darker colors indicate higher weights. The red box indicates the keyphrase to be detected.

4.2 Test without Conversation Context

Although we have shown in the previous section that conversation context is useful for training effective models for keyphrase extraction on microblog posts, it is necessary to consider that conversation context might be unavailable to some microblog posts, which do not sparking any repost or reply message. Under this circumstance, the models trained on messages with conversation context might be affected in extracting the keyphrases for messages without conversation context. To study whether conversation context is critical in testing process, we assume that the conversations are only available for training data, while all the target posts in the test set have no context to be leveraged. To this end, we apply the models trained for the experiment in Section 4.1 on the test posts without using their conversation context. In prediction, context encoders of the trained models take the target posts instead of conversation as input. Results are reported in Table 6, where models with context encoders yield better F1 scores than their counterparts without such encoders no matter providing conversation to test data or not. This observation indicates that encoding conversations in training data helps in learning effective keyphrase extraction models, which is beneficial to detect keyphrases in a microblog post with or without its conversation context. In addition, by comparing Table 6 with Table 4 and 5, we find that, for each model with context encoder, higher F1 scores are observed when conversation context is used in testing process. This observation confirms that, conversation context of target posts helps in indicating keyphrases in prediction.

4.3 Qualitative Analysis

To qualitatively analyze why MemNN encoder generally performs better in comparison, we conduct a case study on the sample instance in Table 1. Recall that the keyphrase should be "*president Duterte*". We compare the keyphrases produced by the joint-layer BiLSTM tagger with various context encoders, given in Table 7. Of all models, only the one with MemNN encoder tags correctly. Interestingly, Avg Emb does not extract any keyphrase. The reason might be that it considers each word in conversations independent and equally important. Therefore, when using this encoder, non-topic words like "*if*" and "*LOL*" may distract the keyphrase tagger in identifying the key information. Models with BiLSTM, Att (BiLSTM), and the basic model without encoder mistakenly extract the sentiment word "*childish*" since sentiment words are prominent on Twitter.

We also visualize context representation generated by MemNN for conversation context in a heatmap shown in Figure 4. It is observed that MemNN highlights different types of words for keyphrases and non-keyphrases. For keyphrases, MemNN highlights topical words such as "*Obama*". For non-keyphrases, MemNN highlights non-topic words, e.g., "*be*", "*to*". Therefore, features learned for keyphrases and non-keyphrases are different, which can thus benefit keyphrase tagger to correctly distinguish keyphrases from non-keyphrases.

4.4 Keyphrases with Various Lengths

To further evaluate our methods, we investigate them on keyphrases with various lengths. Figure 5

	Extracted keyphrase
Gold-standard	*president duterte*
No encoder	*duterte childish*
Context Encoder	
Avg Emb	*NULL*
BiLSTM	*duterte childish*
Att (BiLSTM)	*president duterte childish*
MemNN	*president duterte*

Table 7: Outputs of joint-layer BiLSTM combined with various context encoders given the example illustrated in Table1. "*NULL*": Avg Emb did not produce any keyphrase.

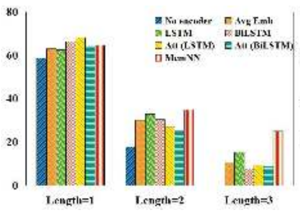
shows the histograms of F1 scores yielded by a single-layer and a joint-layer tagger on Twitter and Weibo when keyphrase lengths are different. Note that we only report the results of BiLSTM taggers because their overall F1 scores are the best according to Table 4 and Table 5.

In general, the F1 scores of all models decrease when keyphrases becomes longer, which implies that detecting longer keyphrases is harder than short ones. In comparison of different context encoders, we observe that MemNN obtained the best F1 score in detection of long keyphrases. This is because MemNN highlights salient content in conversation context by jointly considering its similarities with keyphrases in target posts. When the keyphrases become longer, there are more words in context highlighted, which hence helps keyphrase tagger. For short keyphrases, MemNN is still competitive with other context encoders. The observation suggests that MemNN is robust in detecting various length of keyphrases.

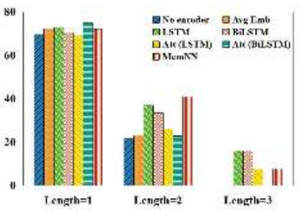
4.5 Error Analysis

In this section, we briefly discuss the errors found in our experiments. It is observed that one major incorrect prediction is additionally extracted neighboring words surrounding a gold-standard keyphrase. For example, in the tweet "*Hillary Clinton accepted gifts from UAE, Saudi Arabia, Oman and others while SOS. CROOKED Podesta Emails 29 ...*", in addition to the gold-standard "*Podesta Emails 29*", our models also extract out "*CROOKED*". In general, these additionally extracted words are mostly modifiers of keyphrases. External features for identifying modifiers can be used to filter these auxiliary parts of a keyphrase.

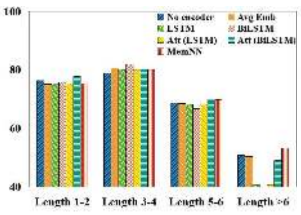
Another main error comes from the words that are not keyphrases in target posts but reflect the topics in conversations. For example, joint-layer



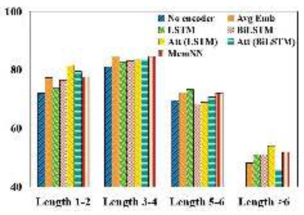
(a) SL BiLSTM on Twitter



(b) JL BiLSTM on Twitter



(c) SL BiLSTM on Weibo



(d) JL BiLSTM on Weibo

Figure 5: Histograms of F1 scores on extracting keyphrases with various lengths. SL BiLSTM: tagger based on single-layer BiLSTM. JL BiLSTM: tagger based on joint-layer BiLSTM. Length: count of words in keyphrases. For each length range, histograms from left to right show the results of No encoder, Avg Emb, LSTM, BiLSTM, Att (LSTM), ATT (BiLSTM), and MemNN.

BiLSTM tagger with MemNN encoder mistakenly extracts "*Hillary*" as a keyphrase for "*DOUBLE STANDARD: Obama DOJ Prosecuted Others For Leaking FAR LESS Than Hillary Espionage URL*" whose keyphrase should be "*Espionage*". Because the corresponding conversation of this post is centered around "*Hillary*" instead of "*Espionage*", such information is captured by the context encoder, which leads to incorrect keyphrase prediction. However, this type of error points out the potential of extending our framework to extracting keyphrases from conversations instead of a post, which would be beneficial to generating summary-worthy content for conversations (Fernández et al., 2008; Loza et al., 2014).

4.6 Ranking-based Models

Table 8 reports the results of ranking models on Twitter and Weibo. We have the following observations. First, tagging-based models perform much better than ranking-based ones in keyphrase extraction. Comparing the results in Table 8 with that in Table 4 and Table 5, all neural taggers outperform non-neural ranking-based models by a large margin. This fact, again, confirms that keyphrase extraction is a challenging task on short microblog messages. Compared to ranking-based models, neural tagging models have the ability

	Twitter			Weibo		
	Pre	Rec	F1	Pre	Rec	F1
w/o context						
TF-IDF	6.3	**48.8**	11.1	1.9	7.3	3.0
TextRank	6.6	18.8	9.7	1.0	8.6	1.7
KEA	3.5	0.8	1.3	0.1	0.2	0.1
w/ context						
TF-IDF	7.9	45.6	13.4	2.1	8.3	3.4
TextRank	4.8	20.8	7.8	1.0	9.5	1.8
KEA	**15.4**	12.9	**14.0**	**2.2**	**12.3**	**3.7**

Table 8: Precision, recall, and F1 scores of ranking-based baselines (%). w/o context: each target post is treated as a document; w/ context: each conversation and its corresponding target post is treated as a document.

to capture indicative features. Second, conversation context improves ranking-based models by a large margin. Simply by aggregating conversations to a pseudo-document, the F1 scores of TF-IDF, TextRank, and KEA are generally better than their counterparts that are only performed on target posts. For TF-IDF and TextRank, which are unsupervised, context remarkably improves recall by enriching more topic-related words. While for supervised method KEA, context improves both precision and recall, because supervision helps in identifying good features from conversations.

5 Related Work

Previous work on extracting keyphrases mainly focuses on formal texts like news reports (Wan and Xiao, 2008) and scientific articles (Nguyen and Kan, 2007). Existing keyphrase extraction models can be categorized as ranking-based models and tagging-based models. Ranking-based methods include models based on graph ranking (Mihalcea and Tarau, 2004; Wan and Xiao, 2008), text clustering (Liu et al., 2009), TF-IDF (Jones, 2004; Zhang et al., 2007; Lee and Kim, 2008; Kireyev, 2009; Wu and Giles, 2013), etc. The empirical study provided by Hasan and Ng (2010) shows that TF-IDF has robust performance and can serve as a strong baseline. Tagging models focus on using manually-crafted features for binary classifiers to predict keyphrases (Frank et al., 1999; Tang et al., 2004; Medelyan and Witten, 2006). Our models are in the line of tagging approaches, and provide an alternative choice that incorporates additionally knowledge from conversations.

Recently, keyphrase extraction methods have been extended to social media texts (Zhao et al., 2011; Bellaachia and Al-Dhelaan, 2012; Marujo et al., 2015; Zhang et al., 2016). These work suffers from the data sparsity issue because social media texts are normally short. Also, they only use internal information in the input text and ignore external knowledge in conversation context. Thus our work provides an improved approach that compensates their limitations.

6 Conclusion

This work presents a keyphrase extraction framework for microblog posts with considering conversation context to alleviate the data sparsity in short and colloquial messages. The posts to be tagged are enriched by conversation context through four types of encoders based on averaged embedding, RNN, attention, and memory networks, which are effective in capturing salient content in conversations that is indicative for keyphrase identification. Experimental results on Twitter and Weibo dataset have shown that by effectively encoding conversation context, our proposed models outperform existing approaches by a large margin. Qualitative analysis confirm that our context encoders capture critical content in conversations.

Acknowledgments

We thank Shuming Shi, Haisong Zhang, Jialong Han, and three anonymous reviewers for their valuable suggestions on different aspects of this work. Chengzhi Zhang was supported by National Social Science Fund of China 17ZDA291.

References

Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. 2014. Neural machine translation by jointly learning to align and translate. *arXiv pre-print* arXiv/1409.0473.

Abdelghani Bellaachia and Mohammed Al-Dhelaan. 2012. NE-Rank: A novel graph-based keyphrase extraction in Twitter. In *Proceedings of the IEEE/WIC/ACM International Conferences on Web Intelligence, WI*. pages 372–379.

Yi Chang, Xuanhui Wang, Qiaozhu Mei, and Yan Liu. 2013. Towards Twitter context summarization with user influence models. In *Proceedings of the Sixth ACM International Conference on Web Search and Data Mining, WSDM*. pages 527–536.

Kyunghyun Cho, Bart van Merrienboer, Çaglar Gülçehre, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv pre-print* arXiv/1406.1078.

Jaeho Choi, W. Bruce Croft, and Jinyoung Kim. 2012. Quality models for microblog retrieval. In *Proceedings of the 21st ACM International Conference on Information and Knowledge Management, CIKM*. pages 1834–1838.

Junyoung Chung, Çaglar Gülçehre, KyungHyun Cho, and Yoshua Bengio. 2014. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv pre-print* arXiv/1412.3555.

Ronan Collobert and Jason Weston. 2008. A unified architecture for natural language processing: deep neural networks with multitask learning. In *Proceedings of the Twenty-Fifth International Conference Machine Learning, ICML*. pages 160–167.

Raquel Fernández, Matthew Frampton, John Dowding, Anish Adukuzhiyil, Patrick Ehlen, and Stanley Peters. 2008. Identifying relevant phrases to summarize decisions in spoken meetings. In *Proceedings of 9th Annual Conference of the International Speech Communication Association, INTERSPEECH*. pages 78–81.

Eibe Frank, Gordon W. Paynter, Ian H. Witten, Carl Gutwin, and Craig G. Nevill-manning. 1999. Domain-specific keyphrase extraction. In *Proceedings of the Sixteenth International Joint Conference on Artificial Intelligence, IJCAI*. pages 668–673.

Debanjan Ghosh, Alexander Richard Fabbri, and Smaranda Muresan. 2017. The role of conversation context for sarcasm detection in online interactions. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*. pages 186–196.

Kevin Gimpel, Nathan Schneider, Brendan O'Connor, Dipanjan Das, Daniel Mills, Jacob Eisenstein, Michael Heilman, Dani Yogatama, Jeffrey Flanigan, and Noah A. Smith. 2011. Part-of-speech tagging for Twitter: annotation, features, and experiments. In *Proceedings of the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 42–47.

Alex Graves. 2013. Generating sequences with recurrent neural networks. *arXiv pre-print* arXiv/1308.0850.

Alex Graves and Jürgen Schmidhuber. 2005. Framewise phoneme classification with bidirectional LSTM and other neural network architectures. *Neural Networks* 18(5-6):602–610.

Aria Haghighi and Lucy Vanderwende. 2009. Exploring content models for multi-document summarization. In *Proceedings of the Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 362–370.

Kazi Saidul Hasan and Vincent Ng. 2010. Conundrums in unsupervised keyphrase extraction: making sense of the state-of-the-Art. In *Proceedings of the 23rd International Conference on Computational Linguistics, COLING*. pages 365–373.

Sepp Hochreiter and Jürgen Schmidhuber. 1997. Long short-term memory. *Neural Computation* 9(8):1735–1780.

Haoran Huang, Qi Zhang, Yeyun Gong, and Xuanjing Huang. 2016. Hashtag recommendation using end-to-end memory networks with hierarchical attention. In *Proceedings of the 26th International Conference on Computational Linguistics, COLING*. pages 943–952.

Karen Spärck Jones. 2004. A statistical interpretation of term specificity and its application in retrieval. *Journal of Documentation* 60(5):493–502.

Kirill Kireyev. 2009. Semantic-based estimation of term informativeness. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 530–538.

Sungjick Lee and Han-Joon Kim. 2008. News keyword extraction for topic tracking. In *Proceedings of the Fourth International Conference on Networked Computing and Advanced Information Management, NCM*. pages 554–559.

Jing Li, Wei Gao, Zhongyu Wei, Baolin Peng, and Kam-Fai Wong. 2015. Using content-level structures for summarizing microblog repost trees. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing, EMNLP*. pages 2168–2178.

Jing Li, Ming Liao, Wei Gao, Yulan He, and Kam-Fai Wong. 2016. Topic extraction from microblog posts using conversation structures. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL*. pages 2114–2123.

Zhiyuan Liu, Peng Li, Yabin Zheng, and Maosong Sun. 2009. Clustering to find exemplar terms for keyphrase extraction. In *Proceedings of the 2009 Conference on Empirical Methods in Natural Language Processing, EMNLP, August, A meeting of SIGDAT, a Special Interest Group of the ACL*. pages 257–266.

Vanessa Loza, Shibamouli Lahiri, Rada Mihalcea, and Po-Hsiang Lai. 2014. Building a dataset for summarization and keyword extraction from emails. In *Proceedings of the Ninth International Conference on Language Resources and Evaluation, LREC*. pages 2441–2446.

Luís Marujo, Wang Ling, Isabel Trancoso, Chris Dyer, Alan W. Black, Anatole Gershman, David Martins de Matos, João Paulo da Silva Neto, and Jaime G. Carbonell. 2015. Automatic keyword extraction on Twitter. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing of the Asian Federation of Natural Language Processing, ACL*. pages 637–643.

Olena Medelyan and Ian H. Witten. 2006. Thesaurus based automatic keyphrase indexing. In *Proceedings of the ACM/IEEE Joint Conference on Digital Libraries, JCDL*. pages 296–297.

Rada Mihalcea and Paul Tarau. 2004. TextRank: Bringing order into text. In *Proceedings of the 2004 Conference on Empirical Methods in Natural Language Processing , EMNLP, A meeting of SIGDAT, a Special Interest Group of the ACL, held in conjunction with ACL*. pages 404–411.

Thuy Dung Nguyen and Min-Yen Kan. 2007. Keyphrase extraction in scientific publications. In *Proceedings of the Asian Digital Libraries. Looking Back 10 Years and Forging New Frontiers, 10th International Conference on Asian Digital Libraries, ICADL*. pages 317–326.

Yuanping Nie, Yi Han, Jiuming Huang, Bo Jiao, and Aiping Li. 2017. Attention-based encoder-decoder model for answer selection in question answering. *Frontiers of IT & EE* 18(4):535–544.

Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. 2013. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 380–390.

Barak A. Pearlmutter. 1989. Learning state space trajectories in recurrent neural networks. *Neural Computation* 1(2):263–269.

Colin Raffel and Daniel P. W. Ellis. 2015. Feedforward networks with attention Can solve some long-term memory problems. *arXiv pre-print* arXiv/1512.08756.

Kenneth Rapoza. 2011. China's Weibos vs US's Twitter: And the winner is? *Forbes* .

Yafeng Ren, Yue Zhang, Meishan Zhang, and Donghong Ji. 2016. Context-sensitive Twitter sentiment classification using neural network. In *Proceedings of the Thirtieth AAAI Conference on Artificial Intelligence*. pages 215–221.

Ricardo Ribeiro, Anatole Gershman, David Martins de Matos, João Paulo Neto, and Jaime G. Carbonell. 2017. Event-based summarization using a centrality-as-relevance model. *Knowledge & Information Systems* 50(3):945–968.

Gerard Salton and Chris Buckley. 1988. Term-weighting approaches in automatic text retrieval. *Information Processing & Management* 24(5):513–523.

SØren Kaae SØnderby, Casper Kaae SØnderby, Henrik Nielsen, and Ole Winther. 2015. Convolutional LSTM networks for subcellular localization of proteins. In *Proceedings of the Second International Conference on Algorithms for Computational Biology*. pages 68–80.

Jie Tang, Juan-Zi Li, Kehong Wang, and Yue-Ru Cai. 2004. Loss minimization based keyword distillation. In *Proceedings of the Advanced Web Technologies and Applications, 6th Asia-Pacific Web Conference, APWeb*. pages 572–577.

Peter D. Turney. 2000. Learning algorithms for keyphrase extraction. *Information Retrieval* 2(4):303–336.

Xiaojun Wan and Jianguo Xiao. 2008. Single document keyphrase extraction using neighborhood knowledge. In *Proceedings of the Twenty-Third AAAI Conference on Artificial Intelligence, AAAI*. pages 855–860.

Jason Weston, Sumit Chopra, and Antoine Bordes. 2015. Memory networks. In *Proceedings of the International Conference on Learning Representations, ICLR*.

Ian H. Witten, Gordon W. Paynter, Eibe Frank, Carl Gutwin, and Craig G. Nevill-Manning. 1999. KEA: Practical automatic keyphrase extraction. In *Proceedings of the Fourth ACM conference on Digital Libraries*. pages 254–255.

Zhaohui Wu and C. Lee Giles. 2013. Measuring term informativeness in context. In *Proceedings of the Human Language Technologies: Conference of the North American Chapter of the Association of Computational Linguistics, NAACL*. pages 259–269.

Hua-Ping Zhang, Hong-Kui Yu, De-Yi Xiong, and Qun Liu. 2003. HHMM-based chinese lexical analyzer ICTCLAS. In *Proceedings of the Second SIGHAN Workshop on Chinese Language Processing*. pages 184–187.

Qi Zhang, Yang Wang, Yeyun Gong, and Xuanjing Huang. 2016. Keyphrase extraction using deep recurrent neural networks on Twitter. In *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP*. pages 836–845.

Yongzheng Zhang, Evangelos E. Milios, and A. Nur Zincir-Heywood. 2007. A comparative study on key phrase extraction methods in automatic web site summarization. *Journal of Digital Information Management* 5(5):323–332.

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 the 49th Annual Meeting of the Association for Computational Linguistics: Human Language Technologies*. pages 379–388.