TR-LDA: A Cascaded Key-Bigram Extractor for Microblog Summarization

Yufang Wu, Heng Zhang, Bo Xu, Hongwei Hao, and Chenglin Liu

***Abstract*—Microblog summarization can save large amount of time for users in browsing. However, it is more challenging to summarize microblog than traditional documents due to the heavy noise and severe sparsity of posts. In this paper, we propose an unsupervised method named TR-LDA for summarizing microblog by cascading two key-bigram extractors based on TextRank and Latent Dirichlet Allocation (LDA). Cascading strategy contributes to a key-bigram set with better noise immunity. Two sentence ranking strategies are proposed based on the key-bigram set. Moreover, an approach of sentence extraction is proposed by merging two ranking results. Compared with some other text content based summarizers, the proposed method was shown to perform superiorly in experiments on Sina Weibo dataset.**

***Index Terms*—Key-Bigram, extraction, microblog summarization, sentence extraction, TR-LDA.**

I. Introduction

Microblog platforms such as Twitter and Sina Weibo have become part of our daily life, from which we can gain information timely to keep in touch with the world every now and then. However, sometimes we may sink into the massive information. A lot of time can be saved for users in browsing if microblog can be summarized automatically. Moreover, text analysis tasks such as classification, clustering and information retrieval can benefit from text summarization due to the reduction of dimensions.

The purpose of this paper is to automatically extract several salient sentences from a set of topic related microblog posts to form a summary to summarize the core contents. From the perspective of traditional document summarization, it can be treated as a multi-document summarization problem by treating each post as a document or a single-document summarization problem by simply concatenating all posts as one document. However, the problem is still more intractable than summarizing any traditional documents, since microblog posts suffer from severe sparsity, heavy noise and bad normalization [1], while traditional documents are usually in nice structure and clear semantic. Most existing microblog summarization methods suffer from low precision.

To overcome the above difficulties, we propose an unsupervised method named TR-LDA to summarize microblog by cascading key-bigram extractors. Unlike most existing methods [1]-[4], which are based on Bag-of-Words (BoW) model to weight sentences or rank sentences directly based on text graph, our TR-LDA method generates summary by two main steps: 1) Extract a key-bigram set to discover the subtopics of the hot topic posts by cascading TextRank and LDA extractors; 2) Rank sentences based on the key-bigram set by two strategies and extract sentences by merging the two ranking results to form a summary. There are two advantages of cascading extractors: 1) TextRank removes the low-ranked bigrams to provide a precise candidate set to the LDA extractor. And a lot of time can be saved when calculating the local density since the size of candidates is cut down. 2) LDA captures the semantic relationship between bigrams so that the centrality of bigrams can be measured more precisely, which contributes to a key-bigram set with less noise finally.

The main contributions of our work can be summarized as follows: 1) We propose an unsupervised key-bigram extracting method by cascading TextRank and LDA extractors. 2) We propose a way to measure bigram centrality by calculating its local density based on the word distribution over topics of LDA model. 3) We propose two efficient sentence ranking strategies and a multi-ranking results merging method for sentence extraction. Therefore, our TR-LDA method can generate precise and high-recall microblog summaries even only based on the text content features. Compared with the existing text content based summarizers, our experimental results on Sina Weibo dataset demonstrate the superiority of our method.

The outline of this paper is as follows. We briefly review the related work in Section II, then describe our proposed TR-LDA method in details in Section III. Experimental results on Sina Weibo dataset are shown in Section IV, followed by a conclusion.

II. Related Work

Text summarization methods can be divided into extractive and abstractive ones. The latter one, re-generating sentences, is good at concentrating contents into a natural summary but highly relies on domain knowledge, thus sufferring from bad popularization. The former one, directly extracting sentences from the texts, is widely used because it is not restricted to text domain and genre. Most extractive summarization tasks are regarded as sentence ranking problems, which can be roughly divided into three types: 1) statistical feature based methods [5], [6], which simply consider term frequency, sentence position and length, title and clue words; 2) lexical chain based methods [7], which construct chains of related words with the help of lexicon such as WordNet, and select strong chains to extract salient sentences according to some standards; 3) graph ranking based methods such as LexRank

Manuscript received October 5, 2014; revised January 7, 2015. This work was supported in part by the National Natural Science Foundation of China (Grants No. 61203281 and No. 61303172).

The authors are with the Institute of Automation, Chinese Academy of Sciences, 95 Zhongguancun East Road, Beijing, 100190, China (e-mail: yufang.wu@ia.ac.cn).

[8] and TextRank [4], which use PageRank [9] to rank the text graph. Nevertheless, traditional text summarization methods are unable to satisfy the need of microblog summarization due to the severe sparsity, heavy noise and bad format of posts. Researches on summarizing microblog is scantly. Features like text contents, social attributes and user influences [10], [11] are widely adopted to summarize microblog. But we only review some summarization methods based on text content that are related to our work here. Sharifi *et al.* [2] proposed Phrase Reinforcement (PR), a graph-based algorithm, to find the most commonly occurring phrases to be included in the summary, which was defeated by a simpler statistical algorithm named Hybrid Term Frequency Inverse Document Frequency (Hybrid TF-IDF) [3] they proposed later. Inouye and Kalita [1] developed Hybrid TF-IDF to generate multiple post summaries by introducing similarity threshold and clustering techniques. One is extracting the most weighted posts directly from all the posts as summary, whose pair-wise similarity is under the similarity threshold. The other is clustering posts before extraction, then extracting posts from different clusters. Surprisely, the former one outperformed other methods, including several well-known traditional text summarizers. However, previous work still suffers from low precision because noise of posts may be introduced when scoring sentences directly based on the whole BoW.

Keyword extraction is closely related to a number of text mining tasks, such as text retrieval, document clustering, classification and summarization. We focus on unsupervised methods for keyword extraction in this paper. TF-IDF [12] is one of the most widely used methods due to its simplicity and efficiency, which simply ranks candidate words by the TF-IDF scores and selects the top N words as keywords. TF-IDF is good at finding new words especially in microblog posts, which usually contain quite a few out-of-vocabulary. However it may fail to extract low-frequency keywords. Graph-based ranking methods, such as TextRank [4] and reinforcement graph [13], [14], have become the state-of-the-art methods for keyword extraction. TextRank executes PageRank [9] on a word graph and ranks words according to the final PageRank scores. Recently, more researches [15], [16] focus on discovering latent semantic relationships between words to reduce vocabulary gap by LDA [17] topic model. LDA models a set of documents, then estimates the topic distribution of each document and the word distribution of each topic, thus words and documents are associated with topics. The semantic similarity between a word and a document can be measured by their topic distributions, which can be used as the ranking score for keyword extraction. Only a handful of summarization work is based on keyword extraction. Ernesto *et al.* [18] exploited a keyphrase extraction methodology to LAKE system at DUC-2005, which chose candidate phrases using linguistic knowledge and extracted keyphrase by TF-IDF term weighting with the position feature. However, the position feature may fail in microblog posts since they are short and in bad format. Some traditional document summarization work also tried to take bigram as lexical unit. Gillick *et al.* [19] and Li *et al.* [20] both summarized traditional multi-documents based on maximizing bigram weights by Integer Linear Programing (ILP). However, no similar work has been applied to the noisy and sparse microblog. Therefore, it still remains to see the performance of summarizing microblog by cascading unsupervised key-bigram extractors.

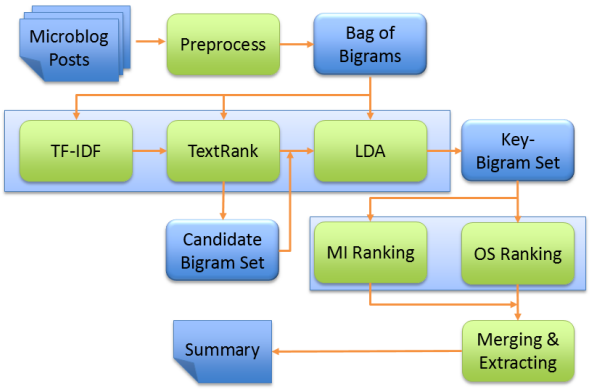


Fig. 1. Framework of the TR-LDA method.

III. The TR-LDA Method

A. Framework

Given a set of microblog posts that are related to the same topic, we extract salient sentences with redundancy removal to form a summary with appropriate length by our TR-LDA method. Fig. 1 shows the whole framework: Firstly, preprocess microblog posts into sentence set, and each sentence is represented as a bag of bigrams. Secondly, extract a key-bigram set by cascading two unsupervised key-bigram extractors: 1) Generate candidate bigram set by TextRank extractor whose text nodes are initialized by TF-IDF values. 2) Extract key-bigrams from the candidate set by LDA based on their local densities. Thirdly, rank sentences by two strategies and extract sentences by merging the two ranking results. Finally, form the extracted sentences into a summary then output it.

B. Preprocessing

In order to make the posts more pure, we remove all topic hashtags, embedded URLs, symbol emotions, forwarding characters and user names. Then we split long posts that consist of more than 30 characters into sentences, and remove short sentences that consist of less than four characters. Consequently, we split sentences into unigrams. Microblog posts usually present two kinds of phenomena: 1) Reposting may lead to massive repeated posts; 2) Users are inclined to express some viewpoints on the same topic with similar or same words and phrases. These repeated words, phrases and sentences mainly deliver some strong views of a topic, which are more likely to be keywords. Moreover, keywords surrounded by other keywords make the sentence more significant. Thereupon, based on the previous preprocessing results, we combine two adjacent unigrams in each sentence as a bigram. Thus each sentence is represented as a bag of bigrams. We take bigram instead of word and phrase as the lexical unit by considering its information and simplicity. Straightforwardly, bigram contains more information than a single word, which is like phrase to some degree. However, it is rather complex to extract syntactic-complete phrase, which

may need extern lexicon and complicated syntactic parsing. We can generate bigram quickly by just concatenating two adjacent words together. Moreover, our ultimate goal is to extract salient sentences to form the summary rather than raw bigrams. Therefore, we care more about the information, rather than whether the format of a bigram is in agreement with the syntactic rule or not.

C. Key-Bigram Extracting Based on TextRank

The basic idea of TextRank is voting or recommendation. Each edge in the graph can be considered as a recommendation. The larger weight of edge means the higher degree of recommendation. Besides, the score of the vertex itself measures the authority of its recommendation. The vertex's authority propagates in the whole graph recursively. Therefore, TextRank computes the salience of a vertex by taking into account the global information from the entire graph recursively, rather than only based on the local vertex-specific information.

Let *V* be the set of vertexes and *E* be the set of edges, we construct a directed weighted graph $G(V,E)$, in which each vertex is a bigram, and each edge is the co-occurring times of two ordered bigrams within a fixed length window (we set to 10). We donate $In(v_i)$ as the set of vertexes that point to vertex v_i, and $Out(v_j)$ as the set of vertexes that pointed by vertex v_j, and w_{ji} as the weight of the edge from v_j to v_i. We can calculate the TextRank [4] score of each vertex as below:

$$S\_TR(v_i) = (1-d) + d * \sum_{v_j \in In(v_i)} \left(w_{ji} * S\_TR(v_j) \Big/ \sum_{v_k \in Out(v_j)} w_{jk} \right), \quad (1)$$

where *d* is the damping factor to keep the model clear of non-convergence when there exists vertex without any successors in the graph. The value of *d* is usually set to 0.85. Although the final scores obtained by TextRank are not sensitive to the initial values, the number of iterations to converge may be different. In order to fasten the converging speed, we initialize the score of bigrams with their TF-IDF values instead of arbitrary values, which can be formally defined as below:

$$S\_TF\_IDF(b_i) = tf(b_i) * \log_2(idf(b_i)), \quad (2)$$

where $tf(b_i)$ is the frequency of bigram occurring in the sentence set, and $idf(b_i)$ is the proportion of the size of the sentence set to the number of sentences that b_i occurs. After running TextRank algorithm, an authority score is associated with each bigram. We rank the bigrams in descending order and select the top r% bigrams as candidate set.

D. Key-Bigram Extracting Based on LDA

If bigrams are grouped into clusters, it is obvious that a bigram cluster usually forms a key aspect of the posts, among which, bigrams with higher centrality (not only the cluster center) seem more reasonable to represent the topic of the cluster. However, we have no need to gain the bigram cluster explicitly if we can obtain the local density of the bigram. Because high centrality in a cluster means high local density within specific distance or specific number of neighbors. Rodriguez *et al.* [21] propose a fast search method to find density peaks. Inspired by their work, we propose to extract key-bigram s from the candidate set by computing their local densities based on the LDA word distribution over topics.

The basic idea of LDA is to discover the latent topics of each document and the relationship between words and topics. We estimate the parameter θ and Φ of LDA by Gibbs Sampling, where $\theta = \left(\theta_1, \cdots, \theta_M\right)^T \in \mathbb{R}^{M \times K}$ is the document - topic matrix, and $\Phi = \left(\Phi_1, \cdots, \Phi_K\right)^T \in \mathbb{R}^{K \times V}$ is the topic - word (it is topic - bigram in our task) matrix. Then we extract key bigrams based on the topic-bigram matrix Φ, in which each column is the distribution of bigram b_v over the *K* topics, and each element $\hat{\varphi}_{k,v}$ is the probability of b_v belonging to topic z_k that measures the importance of bigram b_v in topic z_k to some degree. We normalize Φ along column to gain the standard word distributions over topics. For each bigram in the candidate set, we look up its word distribution in Φ, and calculate its local density by searching for *d* nearest neighbors, where *d* is specified by a ratio of the candidate set size. It is similar to the k-Nearest Neighbors (KNN) algorithm except that we take the average distance from the current candidate bigram to its neighbors to calculate its local density. We use cross entropy to measure the distance since bigrams are represented as probabilistic distribution over topics, which can be formally defined as below:

$$Dis(b_i, b_j) = -\sum_{k=1}^{K} -b_j(k) * \log(b_i(k)) \quad (3)$$

where b_i is the current candidate bigram and b_j is its candidate neighbor. Because we find that taking b_i and b_j as the model and true distribution respectively contributes to better results than the situation in reverse. The more similar the two distributions are, the larger cross entropy they own, since the degree of uncertainty is higher. However, larger value usually means longer distance. So we take the negative value of cross entropy as the distance. Sequentially, the local density of each bigram can be formally defined as below:

$$LocDen(b_i) = -\sum_{b_j \in Ne(b_i, d)} Dis(b_i, b_j) \Big/ d, \quad (4)$$

where $Ne(b_i, d)$ is the *d* nearest neighbors of b_i. Bigram with higher local density are usually more representative and salient. So we rank all the candidate bigrams based on descending order of their local densities. The above extracting process can be summarized in Table I.

E. Sentence Ranking Strategies

A straightforward sentence ranking approach is to give more salience to sentences that contain more key-bigrams, and give less weight to sentences that are too long or too short,

since they are less appropriate to be included in the summary. Concentrating on this idea, we propose two strategies to rank sentences.

TABLE I: THE LDA EXTRACTOR

Input: candidate bigram set and posts

Output: key bigram set

Algorithm:

1. Model posts with LDA and estimate the topic-bigram matrix by Gibbs Sampling. Obtain the standard distribution of each bigram by normalizing the matrix along column.
2. For each bigram b_i in candidate set:
   1) Look up bigram distribution in the bigram-topic distribution matrix.
   2) Calculate the local density:
      a) Initialize a max-heap with capacity of *d* element.
      b) For each bigram $b_j\ (j \neq i)$ in candidate set:
         If the size of max-heap is smaller than *d*, add $Dis(b_i, b_j)$ to the max-heap directly; else if $Dis(b_i, b_j)$ is smaller than the top element of the max-heap, remove the top one and add $Dis(b_i, b_j)$ to the max-heap.
      c) Calculate $LocDen(b_i)$ based on the max-heap.
3. Rank candidate bigrams based on descending order of their local densities, and select the top-N as key-bigrams.

1) Overlap similarity (OS) strategy

OS is a recall-liked score, which counts the overlap bigrams between the sentence and the key-bigram set, divided by the size of key-bigram set. And then dividing the score by the length of sentence can weaken the weight of long sentences. However, some short sentences may gain higher weight on the contrary, which deviates from the idea that too short sentences are not suitable to form the summary. Therefore, when the length of sentence is shorter than the average length of the whole sentence set, we normalize the score by the average length. Formally, OS computes the score of a sentence S_j as follows:

$$S\_OS(S_i) = \frac{\left|\left\{b_i \middle| b_i \in S_j \ \&\ b_i \in KBS\right\}\right|}{\max\left(AveLen, \left|S_j\right|\right) * \left|KBS\right|}, \quad (5)$$

where b_i is the co-occurring bigram, $|S_j|$ is the length of sentence, and $|KBS|$ is the size of key-bigram set.

2) Mutual Information (MI) strategy

The relevance between two variables can be measured by MI. The higher the value is, the tight the two variables are. Thereupon, we can measure what extent a sentence contains the key-bigram set by MI. Sentences with larger MI score should be ranked higher because their higher coverage degree. Formally, MI computes the score of a sentence S_j as follows:

$$S\_MI(S_j) = \sum_{i=1}^{|KBS|} \log\left(\frac{p(b_i, S_j)}{p(b_i)\,p(S_j)}\right) \Big/ \max\left(AveLen, |S_j|\right), \quad (6)$$

where $p(b_i, S_j)$ is the frequency of bigram b_i occurring in sentence S_j, $p(b_i)$ is the frequency of bigram b_i occurring in the sentence set, and $p(S_j)$ is the proportion of the length of sentence S_j to the length of the whole sentence set. We sum up the pairwise mutual information directly without multiplying the joint probability $p(b_i, S_j)$, because most joint probabilities are usually quite small, multiplying them may weaken the distinction of different bigrams. Finally, the score is explicitly normalized by the same normalization factor defined in (5).

F. Sentence Extracting by Merging Ranking Results

OS strategy shows good performance of high recall, while MI strategy obtains better precision. Therefore, it is meaningful to merge the two ranking results when extracting sentences to form the summary. We propose a merge strategy by considering the mean and variance values of two ranks for a specific candidate sentence. Sentence with higher mean rank and low variance in two ranking results should be given higher preference. Because high rank in each ranking result means high recommendation, and low variance in both ranking results indicates the stability of rank. Meanwhile, we also take into account the different weights of two ranking strategies. Consequently, the priority of each candidate sentence after merging can be formally defined as below:

$$Prior(S_i) = \lambda * mean(S_i) + (1-\lambda) * var(S_i), \quad (7)$$

$$mean(S_i) = \delta * ros(S_i) + (1-\delta) * rmi(S_i), \quad (8)$$

$$var(S_i) = \begin{pmatrix} \delta * (ros(S_i) - mean(S_i))^2 \\ +(1-\delta) * (rmi(S_i) - mean(S_i))^2 \end{pmatrix}^{-1/2}, \quad (9)$$

where λ and δ are two parameters with values between 0 and 1, and where $mean(S_i)$ and $var(S_i)$ are the mean and variance ranking values of sentence S_i, for which λ is used to tune the weight of two factors, and where $ros(S_i)$ and $rmi(S_i)$ are the ranks in OS and MI ranking results, for which δ is used to tune the weight of two strategies. Small ranking value means high rank. Hence, sentences with smaller value of selection priority as (7) described should be given higher priority after merging the two ranking results. Moreover, redundancy removal is rather necessary when extracting sentences for summary. Before adding the selected candidate sentence into summary, we compute its similarity with each sentences that already be added in summary as (10) and compare each pairwise similarity with a threshold *t*. We set *t* to 0.5 by considering that a sentence carrying less than 50% new information does not deserve to be included in the summary especially on a big candidate set. Once if the similarity value is larger than the threshold, we give up the current candidate and move to the next one, until we obtain a certain number of sentences to form the summary.

$$Sim(S_i, S_j) = \frac{|\{b_i | b_i \in S_i \& b_i \in S_j|}{\log|S_i| + \log|S_j|} < t. \quad (10)$$

Merging and extracting proceed simultaneously. Merging, whose thought is similar to merge sort, provides candidate sentence for extracting one by one. Once the number of extracted sentences is satisfied, merging and extracting both can be suspended, rather than merge all ranking results firstly. The whole process can be described in Table II.

TABLE II: SENTENCE MERGING AND EXTRACTING

Input: OS and MI ranking results

Output: Summary formed of M sentences

Algorithm:

1. Initialize a space summary set named *SumSet*.
2. Let two pointers *p1* and *p2* point to the beginning of OS and MI ranking results, separately.
3. Calculate the priority of **p1* and **p2* as (7). If *Prior*(**p1*) < *Prior*(**p2*), choose **p1* as candidate sentence, *p1++*; else choose **p2* as candidate sentence, *p2++*.
4. Calculate the similarity of current candidate and all sentences in *SumSet*. If (10) can be satisfied for all pair-wise similarities, add current candidate into *SumSet*, else discard it.
5. If | *SumSet* |< M, go to 3), else go to 6).
6. Combine sentences in *SumSet* to output as a summary.

IV. EXPERIMENTS

A. Experimental Setup

To perform experiments, we crawled a collection of topics from the Sina Weibo hot topic lists.[1] From the topics between May and June of 2013, we selected 50 hot topics that the number of posts of each topic was between 1500 and 2000. Then we invited two volunteers to extract summary for each topic after scanning all the posts of the topic. Each manual summary consisted of 10 sentences from the posts.

Summary evaluation methods can be divided into two types: intrinsic or extrinsic. In extrinsic evaluations, automated summary is measured by how it assists users to perform other tasks such as document retrieval and classification [22]. In intrinsic evaluations, automated summary is evaluated by its content, fluency or grammaticality, which usually compared with manual summaries or directly judged by humans. ROUGE [23], actually a suit of metrics, is the most popular automatic evaluation metric. One of the simplest ROUGE is ROUGE-N, whose basic idea is to measure the co-occurring n-grams between the automated and manual summaries. Let *MS* be the manual summaries, and *AS* be the automated summary. Let $Match(n\_gram)$ be the number of co-occurring n-grams between the manual and automated summaries, and $Count(n\_gram)$ be the number of n-grams in the manual summaries. The Recall, Precision and F-measure of ROUGE-N can be computed as follows:

$$Recall = \frac{\sum_{S \in MS} \sum_{n\_gram \in S} Match(n\_gram)}{\sum_{S \in MS} \sum_{n\_gram \in S} Count(n\_gram)}, \quad (11)$$

$$Precision = \frac{\sum_{S \in MS} \sum_{n\_gram \in S} Match(n\_gram)}{|MS| * \sum_{n\_gram \in S} Count(n\_gram)}, \quad (12)$$

$$F-measure = \frac{2 * Recall * Precision}{Recall + Precision}, \quad (13)$$

where |*MS*| is the number of manual summaries for each topic. We take ROUGE-1 as the evaluation metric to make our work more comparable with Inouye's work [1].

B. Results and Discussions

Some important parameters that may affect the experimental results heavily and comparison of our method with baselines will be discussed in this section. Since LDA model involves random seeding, we compute the average ROUGE-1 results of 50 times to weaken the effects of random seeding. For the ratio of TextRank extractor, large value may increase the burden of LDA extractor, and small value may abandon important bigrams mistakenly. After exhaustive experiments we set it to 30%. For the ratio of nearest neighbors when calculating local density of bigram, Rodriguez *et al.* [21] advises to set the value of it between 1% and 2%. We find the value does not much affect the performance. We set it to 1.4% in the following experiments. For the number of topics in LDA, we find 50 is appropriate after a large number of experiments.

Different sizes of key-bigram set present great impact on the performance of ROUGE-1. The results of recall, precision and F-measure are shown in Fig. 2 by altering the value of size from 75 to 250. As we can see, recall increases along with the increase of size, while precision decreases. And the best F-measure appears at the point where the size of key-bigram set equals to 150. Furthermore, the weights of merging strategy affect the results more significantly. Fig. 3 presents the results of F-measure by changing λ and δ from 0 to 1, separately. As the figure indicates, the values of F-measure present growing trend as a whole, especially when λ is bigger than 0.4. Larger δ means assign more weight to OS-based ranking strategy than MI-based one. Although MI has higher precision than OS, the recall of OS is much better than MI. Synthesizing recall and precision, OS outperforms MI. Therefore, more bias to OS is more likely to get better results. With the cooperation of λ, merge results gain improvements by drawing the advantages of two ranking strategies. When $\lambda = 0.8$ and $\delta = 0.7$, namely give higher weight to the mean rank of two ranking results and OS-based ranking strategy, our TR-LDA method obtain best merging result with F-measure reaching 0.5577.

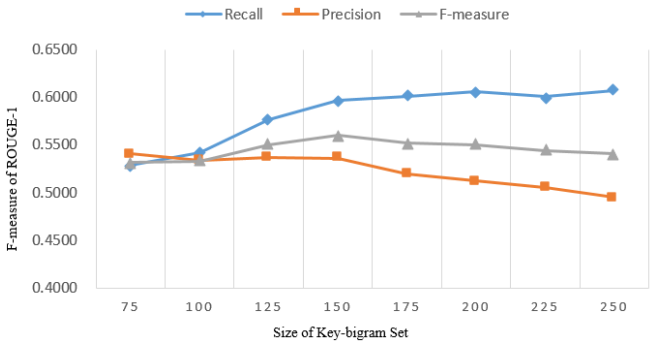
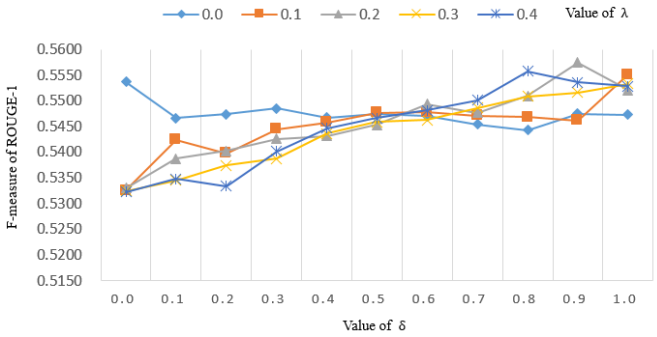
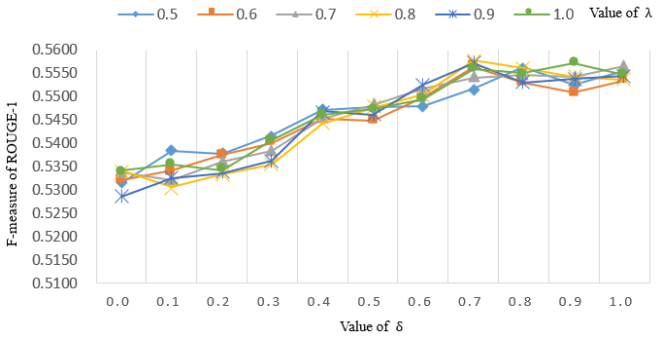


Fig. 2. Performance of different sizes of key-bigram set.

[1] http://huati.weibo.com/



(a) $\lambda \in [0.0, 0.4]$



(b) $\lambda \in [0.5, 1.0]$

Fig. 3. Performance of different weights of merge.

TABLE III: COMPARISON OF BASELINES AND OUR METHOD

Methods	Recall	Precision	F-measure
TextRank	0.5481	0.3354	0.3892
Hybrid TF-IDF	0.4914	0.4303	0.4546
TR-LDA-MI	0.5319	0.5414	0.5325
TR-LDA-OS	0.5954	0.5246	0.5545
TR-LDA-MG	0.5941	0.5337	**0.5577**

We compare our method with two baselines, the Hybrid TF-IDF with similarity threshold summarizer [1] and the TextRank summarizer [4]. The ROUGE-1 performance is shown in Table III, where TR-LDA-MI, TR-LDA-OS and TR-LDA-MG are based on our cascading key-bigram extractors. The only difference is that TR-LDA-MI adopts MI-based ranking strategy only and TR-LDA-OS adopts OS-based ranking strategy only, while TR-LDA-MG merges the two ranking results. We can make some conclusions from the table. 1) Our TR-LDA related methods outperform baselines obviously by gaining more than 8% improvements of F-measure score. 2) The OS strategy is superior to MI strategy according to the F-measure scores. While the former one gets higher recall scores because it is a recall-designed strategy, and the latter one shows better precision values because it penalizes long sentences more severely. 3) Merging two ranking results can improve the F-measure, which maintains the high recall as TR-LDA-OS and improves the precision by introducing MI ranking strategy. 4) TextRank summarizer shows fairly low precision, therefore results in a poor value of F-measure. This proves that it is unwise to directly apply traditional summarizer to summarize short texts such as microblog posts due to their heavy noise and severe sparsity. 5) Hybrid TF-IDF summarizer, which scores sentences based on the whole BoW, synthetically shows much better F-measure score than TextRank while the precision is still not high. Nevertheless, our TR-LDA-based summarizers all outperform Hybrid TF-IDF summarizer by scoring sentences only based on less than 200 bigrams. The main reason can be explained by that most noisy and trivial words are filtered out by the key-bigram set as first, therefore, more precise summaries can be extracted by our methods from noisy microblog posts.

V. CONCLUSIONS

This paper presents an automatic microblog summarization method named TR-LDA by cascading two unsupervised key-bigram extractors. Firstly, by calculating the local density of bigrams based on the word distribution over topics of LDA model, a key-bigram set is extracted from a candidate bigram set that provided by TextRank. Then, sentences are ranked by two strategies based on the key-bigram set. Finally, salient sentences are extracted by merging the two ranking results to form a summary. Compared with the Hybrid TF-IDF summarizer that uses BoW for scoring sentences and the TextRank summarizer that uses direct sentence ranking for summarizing traditional single document, our proposed TR-LDA method yielded superior performance on Sina Weibo dataset.

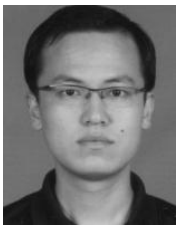
In future work, we plan to combine social attributes to summarize microblog since our method only focuses on text content. Besides, we try to verify the performance of regarding higher n-gram or biterm as lexical unit. Finally, we consider to take advantage of the different weight of key-bigrams when ranking sentences.

REFERENCES

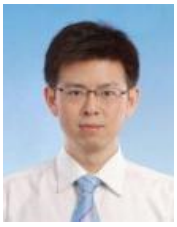
[1] D. Inouye and J. K. Kalita, "Comparing twitter summarization algorithms for multiple post summaries," in *Proc. IEEE 3nd Int. Conf. Social Computing*, 2011, pp. 298-306.
[2] B. Sharifi, M. A. Hutton, and J. K. Kalita, "Summarizing microblogs automatically, in *Proc. HLT/NAACL*, vol.10, pp. 685-688.
[3] B. Sharifi, M. A. Hutton, and J. K. Kalita, "Experiments in microblog summarization," in *Proc. IEEE 2nd Int. Conf. Social Computing*, pp. 49-56, 2010.
[4] R. Mihalcea and P. Tarau, "Text rank: bringing order into texts," *EMNLP*, pp. 404–411, 2004.
[5] H. P. Luhn, "The automatic creation of literature abstracts," *IBM Journal of Research Development*, vol. 2, no. 2, pp.159-165, 1958.
[6] Edmundson, and P. Harold, "New methods in automatic extracting," *Journal of the ACM (JACM)*, vol. 16, no. 2, pp. 264-285, 1969.
[7] Y. I. Song, K. S. Hart, and H. C. Rim, "A term weighting method based on lexical chain for automatic summarization," in *CICLing*, pp. 636-639, 2004.
[8] G. Erkan and D. Radev, "Lexrank: graph-based centrality as salience in text summarization," *Journal of Artificial Intelligence Research*, vol. 22, pp. 457–480, 2004.
[9] L. Page, S. Brin, R. Motwani, and T. Winograd, "The pagerank citation ranking: bringing or-der to the web," Technical Report, Stanford Digital Library Technologies Project, 1998.
[10] Y. Duan, Z. Chen *et al.*, "Twitter topic summarization by ranking tweets using social influence and content quality," *Proceedings of COLING*, 2012, pp. 763-780.
[11] H. Z. Huang *et al.*, "Tweet ranking based on heterogeneous networks," in *Proc. COLING*, 2012, pp. 1239-1256.
[12] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. the Conference on Empirical Methods in Natural Language Processing*, 2003, pp. 216-223.
[13] X. Wan, J. Yang, and J. Xiao, "Towards an iterative reinforcement approach for simultaneous document summarization and keyword extraction," in *Proc. the 45th Annual Meeting of the Association of Computational Linguistics*, 2007, pp. 552–559.
[14] F. F. Liu *et al.*, "Unsupervised approaches for automatic keyword extraction using meeting transcripts," in *Proc. HLT/NAACL*, 2009.

[15] Z. Y. Liu, W. Y. Huang, Y. B. Zheng, and M. S. Sun, "Automatic keyphrase extraction via topic decomposition," in *Proc. the Conference on Empirical Methods in Natural Language Processing*, 2010, pp. 366-376.
[16] W. X. Zhao, J. Jiang, J. He, Y. Song, A. Palakorn, E. P. Lim, and X. M. Li, "Topical keyphrase extraction from twitter," in *Proc. HLT/NAACL*, 2011, vol. 1, pp. 379-388.
[17] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent dirichlet allocation," *Journal of Machine Learning Research*, vol. 3, pp. 993–1022, 2003.
[18] E. D'Avanzo and B. Magnini, "A Keyphrase-based Approach to Summarization: the LAKE System at DUC-2005," in *Proc. DUC*, 2005.
[19] D. Gillick and F. Benoit, "A scalable global model for summarization," in *Proc. Integer Linear Programming for NLP*, ACL, 2009.
[20] C. Li, X. Qian, and Y. Liu, "Using supervised bigram-based ilp for extractive summarization," in *Proc. the 45th Annual Meeting of the Association of Computational Linguistics*, 2013, pp. 1004-1013.
[21] A. Rodriguez and A. Laio, "Clustering by fast search and find of density peaks," *Science*, vol. 344, no. 6191, pp. 1492-1496, 2014.
[22] U. Hahn and I. Mani, "The challenges of automatic summarization," *Computer*, vol. 33, no. 11, pp. 29–36, 2000.
[23] C. Y. Lin and H. Eduard, "Automatic evaluation of summaries using n-gram co-occurrence statistics," in *Proc. HLT/NAACL*, 2003, vol. 1, pp. 71-78.

Yufang Wu received the B.S. degree in automation from Central South University, Changsha, China, in 2012. She is currently pursuing her master's degree in pattern recognition and intelligent systems at the Institute of Automation, Chinese Academy of Sciences (CASIA), Beijing, China. Her research interests include natural language processing especially text analysis, machine learning, pattern recognition and data mining.



Heng Zhang received the B.S. degree in electronic and information engineering from University of Science and Technology of China, Hefei, China, in 2007, and got the Ph.D. degree in pattern recognition and intelligent systems from the Institute of Automation, Chinese Academy of Sciences, Beijing, China, in 2013. Currently, he is an assistant professor at the CASIA. His research interests include short text classification and analysis, speech recognition, handwriting recognition, document analysis and information retrieval.



Bo Xu is currently an associate professor at Institute of Automation, the Chinese Academy of Sciences from July 2011. He received M.E. degree in automation in 2006 from Xi'an JiaoTong University and the Ph.D. degree in pattern recognition and artificial intelligence from the Institute of Automation, Chinese Academy of Sciences, in 2011. His research interests include pattern recognition, image processing, machine learning and especially the applications to character recognition.

Hongwei Hao is currently a professor in the Institute of Automation, Chinese Academy of Sciences (CASIA), China. He received the Ph.D. degree in pattern recognition and intelligent systems from CASIA in 1997. From 1999 to 2011, he worked as an associate professor and then a professor at the University of Science and Technology Beijing, China. From 2002 to 2003, he was a visiting researcher in the Central Research Laboratory, Hitachi Ltd., Tokyo, Japan. His research interests cover large-scale semantic computing, large-scale machine learning theory, and intelligent massive information processing.

Chenglin Liu is a professor at the National Laboratory of Pattern Recognition (NLPR), Institute of Automation of Chinese Academy of Sciences (CASIA), Beijing, China, and is now the deputy director of the laboratory. He received the B.S. degree in electronic engineering from Wuhan University, Wuhan, China, the M.E. degree in electronic engineering from Beijing Polytechnic University, Beijing, China, the Ph.D. degree in pattern recognition and intelligent control from CASIA, Beijing, China, in 1989, 1992 and 1995, respectively. He was a postdoctoral fellow at Korea Advanced Institute of Science and Technology (KAIST) and later at Tokyo University of Agriculture and Technology from March 1996 to March 1999. From 1999 to 2004, he was a research staff member and later a senior researcher at the Central Research Laboratory, Hitachi, Ltd., Tokyo, Japan. His research interests include pattern recognition, image processing, neural networks, machine learning, and especially the applications to character recognition and document analysis. He has published over 180 technical papers at prestigious international journals and conferences. He is on the editorial board of journals Pattern Recognition, Image and Vision Computing, and International Journal on Document Analysis and Recognition. He is a fellow of the IAPR, and a senior member of the IEEE.