IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 15, 2019, accepted December 12, 2019, date of publication December 17, 2019,
date of current version December 31, 2019.

Digital Object Identifier 10.1109/ACCESS.2019.2960285

Detecting Hot Topics From Academic Big Data

BEIBEI WANG[1,2], BO YANG[1,2], SHUANGSHUANG SHAN[1,3], AND HECHANG CHEN[1,4]
[1]Key Laboratory of Symbolic Computation and Knowledge Engineering, Ministry of Education, Jilin University, Changchun 130012, China
[2]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[3]College of Software, Jilin University, Changchun 130012, China
[4]School of Artificial Intelligence, Jilin University, Changchun 130012, China

Corresponding authors: Bo Yang (ybo@jlu.edu.cn) and Hechang Chen (chenhc@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61572226, Grant 61876069, and Grant 61902145, in part by the Jilin Province Key Scientific and Technological Research and Development project under Grant 20180201067GX and Grant 20180201044GX, and in part by the University Science and Technology Research Plan Project of Jilin Province under Grant JJKH20190156KJ.

ABSTRACT Detecting hot topics from massive academic data is a very challenging task. Because various types of academic information are overgrowing, e.g., papers, news, and blogs, which has gone far beyond the limits that researchers can accept. Therefore, how to efficiently and accurately detect hot topics from big academic data is the main problem that researchers are facing. In view of this, we design a general framework for Academic Hot Topic Detection (AHTD). Specifically, in this framework, a DeepWalk-based keyword extraction algorithm for a single paper (S-DWKE) is proposed to detect popular topics in diverse academic fields dynamically. Moreover, we propose a keyword extraction algorithm to extract keywords from multiple articles (M-GCKE), which enables us to detect new topics in emerging academic areas. Then, hot topics can be generated from keywords extracted by the S-DWKE and M-GCKE. A large number of experiments demonstrates the proposed framework effectively improves the performance of hot topic detection in the academic field and performs better than the comparison algorithms. We have applied the above work to the ''Academic Headline'' application to provide the hot topics for researchers.

INDEX TERMS Academic big data, hot spot discovery, keyword extraction, knowledge representation.

I. INTRODUCTION

Innovation is the core driving force for science technology and social development. Keeping up with the latest academic hotspots, constantly discovering new problems, and then proposing new methods to solve them is the only way to maintain academic innovation for researchers. However, according to incomplete statistics, the number of academic papers published worldwide in 2018 has reached more than 3 million. Besides, the number of academic news, blogs, etc. is also very large. Faced with such a huge amount of academic data, how to find hot topics from academic fields is still a challenging problem for all researchers.

Generally speaking, existing studies of hot topics discovery in the academic field can be split into two classes: 1) keywords-based hot topic detection, which mines keywords that can represent topics in academic documents, and then, cluster them to get hot topics; 2) feature-based hot topic detection, which classifies existing topics according to their features to get hot topics [1]. Since we are geared towards a dynamically increasing collection of papers without themes, we adopt the latter. The academic circles has done a great quantity of research on hot topic detection in academic field. For example, using the co-occurrence words network to analyze academic hotspots in the information system [2], using time correlation analysis to identify themes [3], and using the citation papers in the paper to study the structure of academic topics [4]. However, challenges are still faced by these studies from two aspects: first, they cannot identify popular topics from dynamically changing academic data; second, it is difficult to identify keywords in emerging academic fields, as a result, some new academic hotspots cannot be discovered. Finally, we need to consider how to use the extracted keywords to generate hot topics more effectively.

In the well-developed academic fields, researchers publish a lot of papers every year. To track dynamically hot topics of these fields, we take a single paper as a target to extract keywords. In general, keyword extraction methods are generally divided into unsupervised and supervised. Unsupervised methods do not require manual labeling of corpus to assist training, so the accuracy is relatively low. Most of the supervised methods are feature-based traditional

The associate editor coordinating the review of this manuscript and approving it for publication was Haipeng Yao.

This work is licensed under a Creative Commons Attribution 4.0 License. For more information, see http://creativecommons.org/licenses/by/4.0/

machine learning classification algorithms, and these features are statistically obtained from a large amount of data after long-term observation. Therefore, we want to migrate the deep learning method to keyword extraction to automatically learn feature representation of keywords. IPerozzi *et al*. [30] proposed the DeepWalk model, which learns the vector representation of words in a sentence by randomly walking. Inspired by this work, we designed algorithm S-DWKE that construct a co-occurrence network for the words in abstracts and title, and then randomly walk on it to obtain the feature representations. Meanwhile, most keyword extraction methods usually extract keywords from the original text directly. However, the original text often contains many words that have no practical meaning (e.g., the, of), which may affect the accuracy of keyword extraction. Thus S-DWKE contains a step to extract candidate keywords before extracting the real keywords and then extracting the final keywords from candidate keywords.

Due to the short research time of the emerging academic field and the low relevance with other areas, the emerging academic terms are unfamiliar to researchers and are more challenging to capture. To extract emerging keywords more accurately, we propose the algorithm M-GCKE, which expands the relationship between keywords from a single paper to multiple papers. Secondly, to further precisely predict keywords, we use graph convolutional network (GCN) [31] to learn the structural information and node attribute information in the network. Traditional deep learning models do not even apply relationships between nodes to make predictions, and they only use node attributes. However, GCN can perform learning on structure information and node attribute information at the same time, which can make prediction more powerful. A multi-layer perceptron is added to process the GCN output through a nonlinear activation function after one iteration of GCN. The output of the last layer is converted to a probability value by the softmax function.

Hinged on the above insights, we design a novel Hot Topic Detection framework for academic data (AHTD). In this framework, firstly, we propose a based-DeepWalk keyword extraction algorithm for a single paper (S-DWKE). Different from the existing methods, we take a single paper as a target to extract keywords. In the mature-developed academic field, there is a steady stream of papers published every year. When a new paper is published, we only need to extract the keywords from this new paper, then incorporate them into the original hot keywords. Next, we identify the latest hot topics according to combined keywords to realize the dynamic recognition of the hot topics. Secondly, we propose a keyword extraction algorithm based on GCN for multiple papers (M-GCKE), which constructs a co-occurrence network for candidate keywords of multiple papers to extract keywords. In this way, we can obtain emerging keywords that appear less frequently in a single paper but commonly in the multiple, thus discovering some emerging hot topics. Finally, we select hot keywords from extracted keywords and then cluster the hot keywords using their semantic information to generate hot topics. In general, the main contributions of this paper are summarized as below.

- A keyword extraction algorithm based on DeepWalk for a single paper (S-DWKE) is proposed, based on which can dynamically identify hot topics.
- To the best of our knowledge, this work is the first to apply GCN to hot topic detection. A keyword extraction algorithm based on GCN for multiple papers (M-GCKE) is proposed, which enables us to identify hot topics in emerging academic fields effectively.
- An academic hot topic discovery framework (AHTD) is designed. In this framework, based on two keyword extraction algorithms, select popular keywords from the extracted keywords, then use their semantic information to cluster to generate hot topics.

The rest of this paper is organized as follows. In Section II, we briefly introduced the related work, and the overall framework of AHTD is presented in Section III. The experiment and related analysis are given in Section IV. Finally, we summarize and further prospect this paper in Section V.

II. RELATED WORK

A. KEYWORD EXTRACTION

According to the source of the keyword, the existing keyword extraction approaches can fall into two general types. The former is mainly based on candidate keyword extraction [5]–[8]. The Latter is to generate keywords directly from the original text [9]–[12]. The extraction process based on candidate keywords is generally separated into two stages. The first stage extracts candidate keywords from the paper, the second stage scores the candidate keywords and selects the higher scores as the ultimate keywords. Hulth [8] made use of a classifier in machine learning to predict the probability value that of candidate keywords as the final keywords. Shang *et al*. [5] automatically extracted high-quality keywords from large-scale texts without manual labeling. Florescu *et al*. [6] proposed a new feature representation of the phrase to extract keywords. The second category directly uses the model to generate keywords that not included in the paper, Gollapalli *et al*. [7] added a replication mechanism based on the cyclic neural network to generate keywords not found in the paper. Chen *et al*. [9] used relevant restrictions to generate keywords and Zhang and Xiao [10] applied convolutional neural networks (CNN) for keyword generation. Compared with the cyclic neural network proposed by Gollapalli *et al*. [7], CNN can achieve complete parallelism through graphics processing unit hardware facilities.

B. HOT TOPIC DETECTION

Different scholars have different understandings of the definition of hot topics. Chen *et al*. [13] defined the topic that frequent occurrence during specific time intervals as a hot topic. Chen *et al*. [14] considered popular topics as never appearing in the previous time slice, but emerging a large number of documents related to these topics in the current

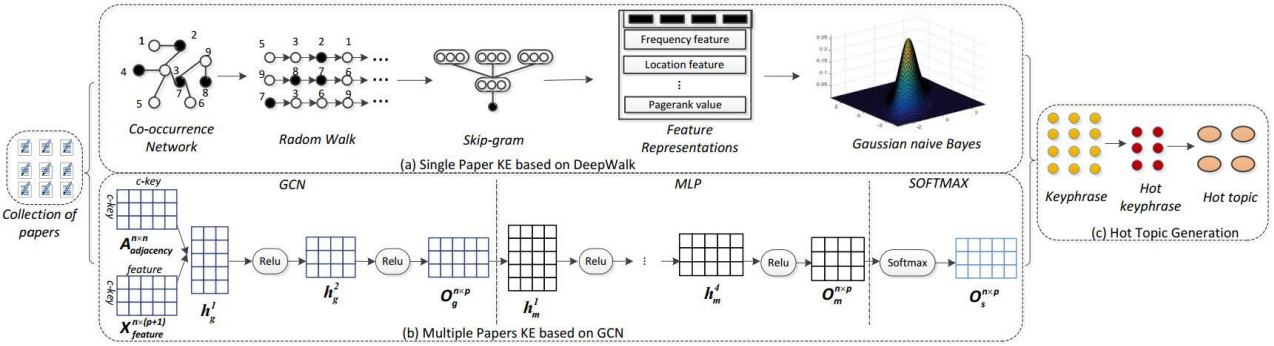


FIGURE 1. The overall framework of AHTD. (a) is to extract the keywords of single paper based on DeepWalk, (b) is to extract the keywords of multiple papers based on Graph Convolutional Network (GCN) and (c) is to generate the hot topics using the keywords extracted by (a) and (b).

time slice. Xie *et al*. [15] believed that the hot topic has a large number of related documents appearing in a period. Until now, researchers give an analysis of the three methods of hot topic detection, including text clustering, topic model, and graph analysis method. Text clustering methods have been extensively used. Yu *et al*. [16] proposed online popular topic detection based on segmentation timeline and age theory. He *et al*. [17] analyzed different topics, life intensity, and trends at different stages built on the popular themes of Weibo, and then built popular themes. Cao *et al*. [18] used Pearson's similarity for top-ranked keywords to identify topics. In the topic model, Yan *et al*. [19] used probabilistic models to solve the problem of the detection of outbreak topics on Weibo. Yan *et al*. [20] proposed the BTM model based on the traditional topic model LDA, then Yan *et al*. [19] improved this model and proposed the BBTM model. Fang *et al*. [23] adopted temporal features to model classification problems for outbreak topics in Twitter, Hammad and El-Beltagy [24] automatically performed burst feature detection from Twitter streams for online topic extraction. Last, in the graph analysis, Katsurai *et al* [21] used the dynamic co-occurrence word network to mine the topic from the academic data, Ma *et al*. [22] applied the graph-based analysis of natural disaster theme mining to discover hot topics.

III. THE PROPOSED FRAMEWORK

In this section, we propose our method Academic Hot Topic Detection framework (AHTD) for academic data. Here we give the definition of the hot topic. The overall framework, as well as each component, will be introduced in detail later.

Definition 1 (Hot Topic): Over a period, topics related to more published papers are called hot topics. A topic consists of many keywords with similar semantics.

Definition 2 (Topic Popularity): There are two sources of information to measure the popularity of the topic: one is the number of keywords included in the topic, and the other is the number of posted papers that related to this topic during that period.

A. THE OVERALL FRAMEWORK

We outline the overall framework of AHTD in Figure 1. In this framework, we get abstracts from the collection of papers as input. Secondly, taking into account the development of different academic fields, we respectively design two methods to extract keywords, as introduced in III-B and III-C. One is the DeepWalk-based keyword extraction algorithm for a single paper (S-DWKE), and the other is the keyword extraction algorithm based on the graph convolutional network (GCN) for multiple articles (M-GCKE). Furthermore, we get hot keywords from the extracted keywords, and then generate hot topics using the hot keywords in Section III-D.

B. THE KEYWORDS EXTRACTION ALGORITHM BASED ON DEEPWALK FOR SINGLE PAPER

In order to dynamically identify hot topics in mature academic fields, we propose a keyword extraction algorithm based on DeepWalk for a single paper (S-DWKE). In S-DWKE, we migrate DeepWalk from the natural language process to keyword extraction, which constructs a co-occurrence network for the words in the title and abstract, and then randomly walks on it to automatically learn the feature representation of the candidate keywords. In addition, we quoted six already designed features and used them together with feature representations for keyword prediction. In S-DWKE, we get the abstracts from a collection of papers as input and output the keywords of each paper. Figure 2-(a) gives the structure of the S-DWKE. We will introduce each composition in detail later.

1) CANDIDATE KEYWORD EXTRACTION

Generally speaking, candidate keywords are the potential keywords of a paper, which exist in the title and abstract in the form of a combination of nouns and adjectives. Moreover, they may be proper nouns that in the way of a single word, and that of all letters are uppercase. Therefore, we design a specific algorithm to extract candidate keywords from the title and abstract of a paper, as shown below:

In this algorithm, we initialize the candidate keywords $\mathcal{K}_c$, then delete the punctuation in the title $\mathcal{P}_t$ and abstract $\mathcal{P}_a$, and then assign them to $\mathcal{P}'_t$ and $\mathcal{P}'_a$. Right after, tokenize text of $\mathcal{P}'_t$ and $\mathcal{P}'_a$ to list of words and assign them to W. Next, a regular expression $\mathcal{G}$ is customized according to the composition of candidate keywords, which represents a phrase that starts with

Algorithm 1 Candidate Keywords Extraction Algorithm

Input: title $\mathcal{P}_t$ and abstract $\mathcal{P}_a$ of a paper $\mathcal{P}$;
Output: candidate keywords $\mathcal{K}_c$ of paper $\mathcal{P}$;

1: $\mathcal{K}_c \leftarrow initialize$;
2: $\mathcal{P}'_t, \mathcal{P}'_a \leftarrow remove_punctuation(\mathcal{P}_t, \mathcal{P}_a)$;
3: $W \leftarrow word_tokenize(\mathcal{P}'_t, \mathcal{P}'_a)$;
4: $\mathcal{G} \leftarrow r$'KT:{(<JJ>*<NN.*>)?<JJ>*<NN.*> +}';
5: $M \leftarrow$ *parse all phrases matching* $\mathcal{G}$ *from* W;
6: **for** $m \in M$ **do**
7: **if** $((m.words) \geq 2$ ***and*** $m \notin \mathcal{K}_c)$ **then**
8: $\mathcal{K}_c \leftarrow \mathcal{K}_c \cup \{m\}$;
9: **if** $((m.words) == 1$ ***and*** $(m.isupper())$ ***and*** $m \notin \mathcal{K}_c)$ **then**
10: $\mathcal{K}_c \leftarrow \mathcal{K}_c \cup \{m\}$;

an adjective or a noun and ends with a noun. After that, we use $\mathcal{G}$ to match a group of phrases from W and assign to M. Finally, in lines 6 to 10, we find phrases of two or more words and professional terms of a single word from the candidate phrases M as final candidate keywords.

2) LEARNING CANDIDATE KEYWORD FEATURE REPRESENTATION

We learn feature representations of candidate keywords based on DeepWalk in this paper. Therefore, a co-occurrence words graph has to be constructed before that. Firstly, identifying adjectives and nouns $\mathcal{D}$ from target paper's title $\mathcal{P}_t$ and abstract $\mathcal{P}_a$. Secondly, building an undirected graph $G(V, E)$, the every word $d \in \mathcal{D}$ corresponding to a vertex $v_i \in V$, if v_i and v_j occur together in k continuous words, an edge (v_i, v_j) is established between v_i and v_j. Further, the weight w_{ij} on edge (v_i, v_j) is equal to the number of co-occurrence of v_i and v_j within k continuous words in the same paper, and we set k to 3.

In this step, the graph $G(V, E)$ generated by the above step as input, our goal is to map each vertex $v_i \in V$ to a low-dimensional feature space. S-DWKE is a method of applying the language model to keyword extraction, so analogy with language model, our input includes a corpus and vocabulary. All the vertexes V in graph G make up the vocabulary, and in order to construct the corpus, we perform a random walk strategy with bias. Suppose, starting from arbitrary vertex $v_i \in V$, perform a random walk with bias and generate a random walk sequence $v_{i_1}, v_{i_2}, \ldots, v_{i_\sigma}$, the length σ obeys the following probability distribution.

$$\alpha = P(v_{i_j} = x | v_{i_{j-1}} = y) = \begin{cases} \frac{w_{xy}}{S}, & \text{if } (x, y) \in E \\ 0, & \text{otherwise} \end{cases} \quad (1)$$

In the (1), the w_{xy} is the weight on the edge $(x, y) \in E$, and sum the weights on all edges of the graph G to obtain S. From above, we sample a fixed bias for each vertex according to the weight of edges and then perform a random walk, which results in a sequence of vertex as a corpus. Corpus and vocabulary together as input to learn the feature embedding

Algorithm 2 Generate the Feature Embedding of the Candidate Keyword

Input: the co-occurrence words graph $G(V, E)$ and candidate keywords $\mathcal{P}_{\mathcal{K}_c}$;
Output: the feature embedding $\boldsymbol{\mathcal{E}}_{\mathcal{K}_c}$ of $\mathcal{P}_{\mathcal{K}_c}$;

1: $\boldsymbol{\mathcal{E}}_{\mathcal{K}_c}, walks \leftarrow initialize$;
2: **for** $i \leftarrow 0$ to num_walks **do**
3: **for** $v \in V$ **do**
4: $path \leftarrow [v]$;
5: **while** $len(path) < walk_length$ **do**
6: $cur \leftarrow path.last$;
7: $cur_neig \leftarrow list(G.neighbors.(cur))$;
8: $weights \leftarrow get_weights(cur)$;
9: $n \leftarrow cur_neig[weighted_choice(weights)]$;
10: $path \leftarrow path \cup n$;
11: $walks \leftarrow path \cup walks$;
12: $my_corpus \leftarrow build_corpus(walks)$;
13: $\boldsymbol{\mathcal{E}}_{\boldsymbol{words}} \leftarrow Word2Vec(my_corpus, V)$;
14: **for** $p \in \mathcal{P}_{\mathcal{K}_c}$ **do**
15: $\mathcal{Q} \leftarrow p.split()$;
16: **for** $q \in \mathcal{Q}$ **do**
17: $\boldsymbol{\mathcal{E}}_{\boldsymbol{q}} \leftarrow \boldsymbol{\mathcal{E}}_{\boldsymbol{words}}.get(q)$;
18: $\boldsymbol{\mathcal{E}}_{\boldsymbol{\mathcal{Q}}} \leftarrow \boldsymbol{\mathcal{E}}_{\boldsymbol{\mathcal{Q}}} + \boldsymbol{\mathcal{E}}_{\boldsymbol{q}}$;
19: $\boldsymbol{\mathcal{E}}_{\boldsymbol{p}} \leftarrow \boldsymbol{\mathcal{E}}_{\boldsymbol{\mathcal{Q}}} / the\ number\ of\ words\ in\ q$;
20: $\boldsymbol{\mathcal{E}}_{\mathcal{K}_c} \leftarrow \boldsymbol{\mathcal{E}}_{\mathcal{K}_c} \cup \boldsymbol{\mathcal{E}}_{\boldsymbol{p}}$;

representation of vertex. Right after, take the average vector of words that construct this candidate keyword as its feature representation. The exact algorithm is shown in Algorithm 2.

In this algorithm, *walks* is the set of random walk sequences, *num_walks* presents the number of walks to be sampled for each node, and we set it to 100. *walk_length* is the length of a random walk, which is set to 10. The main task of 4-10 lines is to generate the random walk sequences of node v. We start a random walk from node v and add it to the current random walk sequence *path*. If the length of *path* is less than the *walk_length*, we get the last node *cur* of *path*, all neighbor nodes *cur_neig* of *cur* and the weights *weights* on all the adjacent edges of *cur*. Next, we select randomly a neighbor n based on *weights* from the neighbor nodes *cur_neig* and add it to the *path*. We generate a path for each node to walk randomly and store it in the *walks*. After 100 walks to be sampled for each node, we use *walks* to generate the corpus *my_corpus*, and then get the feature representation $\boldsymbol{\mathcal{E}}_{\boldsymbol{words}}$ of nodes in G with the vertex V and the *my_corpus* as input. In lines 14 to 20, we split each candidate keyword $p \in \mathcal{P}_{\mathcal{K}_c}$ into words $\mathcal{Q}$, add the embedding $\boldsymbol{\mathcal{E}}_{\boldsymbol{q}}$ of word $q \in \mathcal{Q}$, then average them to get the feature vector $\boldsymbol{\mathcal{E}}_{\boldsymbol{p}}$ of the corresponding candidate keyword p. In the Word2Vec method, the training model we adopted is Skipgram [29]. The relevant hyperparameters are set as follows: sentences are the corpus generated by random walks as the above step, the alpha is set to 0.09, the size is set to 300, hs is 1, sg is 0,

workers are 4, and min_count is 0. The other parameters are the default parameters in Anaconda3.[1]

3) OTHER FEATURES FOR GENERATING KEYWORDS

The features of candidate keywords, in addition to the feature representation, we add some additional features, as shown below.

a. Frequency of occurrence f_1. Let $\mathcal{K}_c$ be an arbitrary candidate keywords, $[w]$ are the words that construct the candidate keywords $\mathcal{K}_c$, $Count_{w_j}$ is the count of times that $w_j \in [w]$ appears in the paper, and $N_{\mathcal{K}_c}$ is the number of words in $[w]$.

$$f_1 = \sum_j Count_{w_j}/N_{\mathcal{K}_c} \tag{2}$$

b. Location that first appears in the paper f_2. w_1 is the first word in $[w]$, $Location_{w_1}$ is the location that w_1 first appears in the paper. T_w is the number of words in the paper.

$$f_2 = Location_{w_1}/T_w \tag{3}$$

c. Propagation frequency f_3. Let d be the number of words in the between the position where the candidate keyword appears first and the appears last.

$$f_3 = d/T_w \tag{4}$$

d. Length. The number of words contained in $\mathcal{K}_c$ is the length $N_{\mathcal{K}_c}$.

e. Frequency in Wikipedia f_4. The frequency of the candidate keyword in Wikipedia is defined as a vector. The value of each dimension of the vector indicates the frequency of occurrence of words in the candidate keyword in Wikipedia. Let p be the frequency of $w_j \in [w]$ appearing in Wikipedia, and q is the frequency of "the" appearing in Wikipedia.

$$f_4 = p/q \tag{5}$$

f. Impact PageRank value f_5. An undirected graph that constructed from words, which can get the PageRank value of each vertex, and then get the PageRank value of each candidate keyword. Let $Page_{w_j}$ be the PageRank value of w_j in $[w]$ of which construct the candidate keyword $\mathcal{K}_c$.

$$f_5 = \sum_j Page_{w_j} \tag{6}$$

4) GENERATING KEYWORDS FROM CANDIDATE KEYWORDS

Up to now, we have got candidate keywords and some of their features. Next, we use the Gaussian Naive Bayes algorithm to obtain the ultimate keywords from candidate keywords. Firstly, we need to make a label for each candidate keyword. In general, we can get the real keywords of a paper after the abstract, if a candidate keyword appears in the real keywords, we make the label as 1, otherwise 0. Then during training, the input is the candidate keywords $\mathcal{K}_c$, their features F and labels L. When testing, the input is candidate keywords $\mathcal{K}_c$ and their features F, the output is the probability π that $\mathcal{K}_c$ belong to the keywords of this paper. At last, candidate keywords of probability top-5 are selected as the keywords of this paper. The exact algorithm is shown below:

[1]https://www.anaconda.com/distribution/

Algorithm 3 An Algorithm for Generating Keywords

Input: candidate keywords $\mathcal{K}_c$ and features F;

Output: The probability π that each candidate keyword belongs to a paper;

1: *Model = GaussianNB(); //definition classifier*
2: *Model.fit(features,labels); //automatically fit the features and labels of the training set*
3: *Model.predict_proba($\mathcal{K}_c$,F); //forecast with test set*

C. THE KEYWORD EXTRACTION ALGORITHM BASED ON GCN FOR MULTIPLE PAPERS

The keyword extraction algorithm S-DWKE is for a single paper, ignoring the relationship of keywords in different papers, which may lead to missing some relevant keywords. For example, in the emerging academic field, some specialized terms may appear only once in a single paper, but multiple times in multiple papers. Therefore, we propose a keyword extraction based on Graph Convolutional Network (GCN) algorithm for multiple papers (M-GCKE), which builds a graph for candidate keywords of multiple papers to extract keywords. When extracting keywords for the emerging academic field, the accuracy can be significantly improved. The main frame of the M-GCKE algorithm is shown in Figure 2-(b), which will be described in detail later.

1) CANDIDATE KEYWORDS CO-OCCURRENCE GRAPH CONSTRUCTION

Similar to the construction of Word Co-occurrence Graph, we build a co-occurrence network $G'(V,'E')$ for candidate keywords. Now we have obtained candidate keywords $\mathcal{K}_c$ of multiple papers using Algorithm 1, each of which corresponds to a vertex $v' \in V'$. If candidate keyword v'_i and v'_j appear in the same paper, we connect an edge $(v'_i, v'_j) \in E'$ between them.

2) GRAPH NEURAL NETWORK INPUT DATA DESCRIPTION AND PROCESSING METHOD

The input of M-GCKE contains three parts, an adjacency matrix $A^{n\times n}$, a feature matrix $X^{n\times(p+1)}$ and a label matrix $L^{n\times 1}$. The matrix A describes the co-occurrence relationship among candidate keywords, which is G'. The matrix X describes the attribute information of the candidate keywords. We take into account two kinds of attribute information, the first is the number of times a candidate keyword appears in all papers, and the second is the first occurrence of the candidate in each paper. The label matrix L, which records whether the candidate keywords appear in the real keywords, marked as 1 if it appears, 0 otherwise. The multi-layer neural

network propagates according to the propagation rule of formula (7) [31] and obtains the feature embedding representation of the vertex in the graph in each layer.

$$H^{l+1} = \sigma(\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}H^{-1/2}W^{(l)}) \tag{7}$$

In the formula(8), $\hat{A}$ is the sum of A and the identity matrix I. The diagonal element of degree matrix $\hat{D}$ is the sum of the elements of each column of $\hat{A}$ as shown in formula (9), $W^{(l)}$ is a layer-specific trainable weight matrix.

$$\hat{A} = A + I \tag{8}$$

$$\hat{D}_{ii} = \sum_j \hat{A}_{ij} \tag{9}$$

σ is the activation function and the Relu is usually chosen. H^l is vector representation of nodes in the l^{th} layer, $H^0 = X$.

3) GENERATING KEYWORDS FORM CANDIDATE KEYWORDS

As shown in Figure 2-(b), the feature embedding $\boldsymbol{X^{n\times(p+1)}}$ and adjacency matrix $A^{n\times n}$ as the input at the first layer of GCN, then output a feature embedding $\boldsymbol{O^{n\times 16}}$ through the hidden layer vector $\boldsymbol{h}_g^1 \in \mathbb{R}^{d_x\times d_h}$ and set its dimension to 16. At the second layer, take the output $\boldsymbol{O^{n\times 16}}$ of the first layer as input, then output the feature vector $\boldsymbol{O^{n\times p}}$ through the hidden vector $\boldsymbol{h}_g^2 \in \mathbb{R}^{d_x\times d_h}$, and the dimension of that is set to p, which is the number of papers. After one iteration of the GCN, a Multi-Layer Perceptron is added to process the output of GCN through a nonlinear activation function. Finally, we adopt softmax to convert the output of the last layer to probability values, then select candidate keywords of probability top-5 as the final keywords for each paper. Our loss function, such as formula (10):

$$Loss = -\sum_i^N y_i \log \hat{y}_i + (1 - y_i)\log(1 - \hat{y}_i) \tag{10}$$

where $y_i \in L^{n\times 1}$ records whether the candidate keyword $\mathcal{K}_{c_i}$ appears in the real keywords of a paper, $\hat{y}_i$ is our predicted value, and N is the number of predicted final keywords. The exact algorithm is as shown in Algorithm 4.

Algorithm 4 Generate Keywords Using GCN

Input: $A^{n\times n}$, $\boldsymbol{X^{n\times(p+1)}}$ and labels $L^{n\times 1}$;
Output: The probability η that each candidate keyword belongs to a paper;

$\hat{A} \leftarrow$ *Normalize matrix A according to formula (8)*;
$\bar{A} \leftarrow \hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$; *//$\hat{D}$ is calculated according to formula (9)*
$H_1 \leftarrow Relu(\bar{A}XW_1 + b_1)$; *// first layer*
$H_2 \leftarrow Relu(\bar{A}H_1W_2 + b_2)$; *// second layer*
$H_3 \leftarrow Relu(H_2W_3 + b_3)$; *// Multi-Layer Perceptron*
$H_4 \leftarrow Relu(H_3W_4 + b_4)$;
$H_5 \leftarrow Relu(H_4W_5 + b_5)$;
$H_6 \leftarrow Relu(H_5W_6 + b_6)$;
$Y_{pre} \leftarrow Softmax(H_6)$; *// convert to probability*
$Optimizer \leftarrow AdamOptimize(learning_rate)$;
$Loss \leftarrow masked_softmax_cross_entropy(Y_{pre}, L)$;

Algorithm 5 Generate Hot Topics Using Hot Keywords

Input: 100 hot keywords K_h;
Output: k hot topics Φ;

if K_h *are in Emglish* **then**
    load the vector representation $\boldsymbol{\mathcal{E}_{trained}}$ *of the words in GoogleNews-vectors- negative300.bin.gz*;
else
    load the vector representation $\boldsymbol{\mathcal{E}_{trained}}$ *of the words in newsblogbbs.vec*;
$\boldsymbol{\mathcal{E}_{K_h}} \leftarrow initialize$;
for $k \in K_h$ **do**
    $[w] \leftarrow k.split()$;
    $\boldsymbol{\mathcal{E}_w} \leftarrow initialize$;
    for $w_i \in [w]$ **do**
        $\boldsymbol{\mathcal{E}_{w_i}} \leftarrow \boldsymbol{\mathcal{E}_{trained}}.get(w_i)$;
        $\boldsymbol{\mathcal{E}_w} \leftarrow \boldsymbol{\mathcal{E}_w} + \boldsymbol{\mathcal{E}_{w_i}}$;
    $\boldsymbol{\mathcal{E}_{K_h}} \leftarrow \boldsymbol{\mathcal{E}_{K_h}} \cup \boldsymbol{\mathcal{E}_w}$;
$n \leftarrow 10$;
$\Phi \leftarrow$ *enter* $\boldsymbol{\mathcal{E}_{K_h}}$ *and k to K-Means*;

D. HOT TOPIC DETECTION FOR ACADEMIC DATA

The more relevant papers on a topic, the hotter the topic is, which can be seen from the definition of hot topics. Besides, a hot topic consists of several similar semantics keywords. So we think a hot topic is made up of hot words, and hot words are that are included in more papers. Therefore, according to the number of papers interrelated with each keyword, we select the top 100 as popular keywords. After that, we use the K-Means algorithm to cluster according to their semantic information to generate hot topics. Suppose $C_1, C_2, \ldots, C_k$ are k topics we generate, μ_i is the mean vector of C_i, our goal is to minimize the squared error E:

$$E = \sum_{i=1}^{k}\sum_{x\in C_i}\|x - \mu_i\|_2^2 \tag{11}$$

$$\mu_i = \frac{1}{|C_i|}\sum_{x\in C_i} x \tag{12}$$

The specific algorithm for generating hot topics is shown as follows:

In this algorithm, googlenews-vecctors-negative300.bin is the word vector of 300-dimensional news corpus pre-trained by Google with word2vec, and newsblogbbs.vec is a pre-trained Chinese word vector. If the hot keywords are English, we will use the former word vector, otherwise the latter. For each hot word $k \in K_h$, we split it into single words $[w]$, get the vector of $wi \in [w]$ from $\boldsymbol{\mathcal{E}_{trained}}$, and then add them as the feature vector of the hot word k. Finally, input all the hot word vectors $\boldsymbol{\mathcal{E}_{K_h}}$ and the number of categories n into K-Means to get hot topics.

IV. EXPERIMENTS AND ALGORITHM APPLICATION

In this section, we first introduce the experiment and analysis of two different keyword extraction algorithms in detail, and then introduce the experiment and application of the hot topic discovery algorithm.

A. DATASETS AND METRICS

For the sake of evaluating the effectiveness of the algorithms, we use four public datasets and a manually constructed dataset. Each paper in the datasets contains the title, abstract, and keywords. The details of these datasets are described as follows.

- Clu_Com. This is a manually constructed dataset using a web crawler, which contains 2792 papers published recently by Clustering Computer,[2] including the title, abstract, and keywords of the paper.
- WWW [25]. This dataset is provided by Gollapalli and Caragea, which contains titles, abstracts, and keywords of a total of 1,329 papers.
- SemEval [26]. The dataset contains 244 papers in total.
- Krapivin [27]. This dataset contains 2304 papers and keywords provided by the authors.
- Inspec [8]. This dataset contains abstract information for 2000 papers in total.

For the evaluation metrics of keyword extraction algorithm and hot topic detection algorithm, we use precision P, recall R and F1-Score F_1 to measure the experimental effect. The formulas for calculating the three evaluation metrics are as follows. Assuming that the predicted keywords for papers are X, and the real keywords is Y.

$$H = X \cap Y \tag{13}$$

$$P = H/X \tag{14}$$

$$R = H/Y \tag{15}$$

$$H = \frac{2PR}{P+R} \tag{16}$$

Remark: In the hot topic detection algorithm, the predicted hot topics are denoted as X, and the real hot topics are (generated by the hot topic detection algorithm using the actual keywords in the paper) represented as Y.

B. PERFORMANCE FOR KEYWORDS EXTRACTION BASED ON DEEPWALK OF SINGLE PAPER

To evaluate the performance of the proposed algorithm, we conduct comparative experiments against the following method, including SurfKE [6] and TextRank [28]. SurfKE is a method proposed by a paper from AAAI in 2018, we have made improvements here, so we use it as a comparison algorithm. And TextRank as a baseline to be compared. Table 1 lists the datasets we use.

For each paper in different datasets, we select five candidate keywords with the highest probability as the final predicted keywords. Table 2 gives the experimental results of various algorithms, from which we can see that the results of our algorithm on the four datasets Clu_Com, WWW, Krapivin and Inspec are better than the other two algorithms. The experimental analysis is as follows. On the one hand, SurfKE believes that candidate keywords are phrases constructed with two or three nouns and adjectives in succession and that the last word is not an adjective. However, this will produce semantically similar candidate keywords, which will result in duplicate keywords. More seriously, they ignore some of the more significant proper nouns, each word of which is in uppercase. Therefore, our algorithm avoids producing candidate keywords with semantic duplication and adds proper nouns in the corresponding fields to candidate keywords. On the other hand, in the feature selection of candidate keywords, SurfKE only considers the feature vector representation generated by the random walk. In addition to this, we take into account simultaneously the features such as the frequency of occurrence, location that first appears in the paper, propagation frequency, length, frequency in Wikipedia, and Impact PageRank value. We combined the multiple features of the candidate keywords for training, results in the test dataset improved significantly compared to SurfKE.

TABLE 1. Statistics of the dataset for S-DWKE.

Dataset	Clu_Com	WWW	SemEval	Krapivin	Inspec
#train	2100	1329	144	1611	1500
#test	692	1064	100	693	500
#items	2792	265	244	2304	2000

C. PERFORMANCE FOR KEYWORDS EXTRACTION BASED ON GCN OF MULTIPLE PAPERS

We randomly select 100 papers for each dataset as our experimental data. Each paper extracted 24.5 candidate keywords on average, so a total of 2450 candidate keywords are used to construct the co-occurrence graph. Among the 2,450 candidate keywords, we select the 1/3 of them as the train set, verification set, and test set, respectively. On average, each dataset has 817 nodes to build a co-occurrence graph. Consistent with S-DWKE S-DWKE, we will select the five candidate keywords with the highest probability for each paper as its final keywords. The difference in accuracy between the M-GCKE and the other three algorithms can be clearly seen from Figure 2.

Table 2 shows the keyword prediction results of all algorithms, from which we can see that our algorithm has a better experimental effect on the four data sets Clu_Com, SemEval, Krapivin and Inspec. In the M-GCKE algorithm, we input an adjacency matrix A and a feature matrix X into a two-layer GCN model, which propagates twice in the forward process and performs convolution effectively for two-order neighbors of each node. Compared to S-DWKE, it learns a better feature representation. Why? Firstly, as shown in equation (8), GCN obtains $\hat{A}$ by adding an identity matrix I to A, which can

[2]https://link.springer.com/journal/10586

TABLE 2. Experimental results of different keyword extraction methods.

Dataset	Clu_com			SemEval			WWW			Krapivin			Inspec		
Algprithm	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)	P(%)	R(%)	F1(%)
TextRank	3.8	4.0	3.8	4.5	6.6	5.2	4.3	4.1	3.8	5.0	5.4	4.9	10.5	6.6	7.6
SurfKE	8.0	9.2	8.4	6.6	9.1	7.4	8.4	8.8	8.1	11.4	11.3	10.7	24.5	14.8	17.3
S-DWKE	12.5	13.7	12.8	5.9	8.8	6.9	**10.2**	**10.6**	**9.8**	15.6	13.7	13.6	38.9	22.8	27.3
M-GCKE	**16.0**	**17.7**	**16.7**	**7.6**	**9.9**	**8.4**	6.2	6.9	6.1	**17.8**	**14.1**	**13.7**	**44.6**	**23.7**	**29.7**

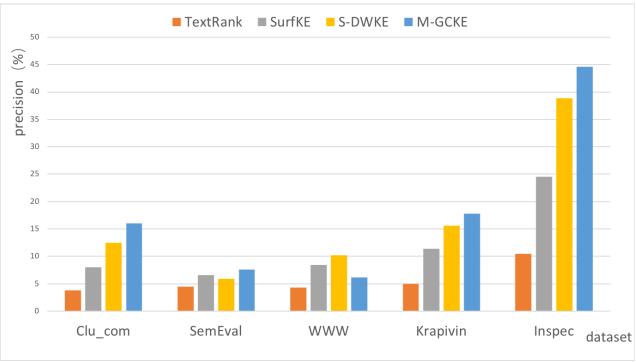


FIGURE 2. Experimental results of keywords extraction using different methods.

TABLE 3. Experimental results of two keyword extraction algorithms without co-occurrence relations.

Dataset	Algorithm	P(%)	R(%)	F1(%)
Clu_Com	S-DWKE	13.9	15.7	14.5
	M-GCKE	14.8	17.2	15.7
SemEval	S-DWKE	15.9	8.8	6.9
	M-GCKE	7.0	9.1	7.7
WWW	S-DWKE	8.3	9.5	8.5
	M-GCKE	7.1	7.3	6.9
Krapivin	S-DWKE	10.3	12.3	12.3
	M-GCKE	11.5	14.1	13.7
Inspec	S-DWKE	45.1	23.7	29.7
	M-GCKE	44.6	23.7	29.7

not only get the feature vector of the neighbor nodes but also integrate the feature vector of the node itself. Secondly, GCN further normalizes $\hat{A}$ by $\hat{D}^{-1/2}\hat{A}\hat{D}^{-1/2}$, avoiding changing the original distribution of features when multiplying with feature matrix X. Therefore, as shown in equation (7), the propagation rules can be interpreted as differentiable and parameterized variants of hash functions used in the original Weisfeiler-Lehman (WL) [32] algorithm. Since the model is parameterizable, we add features to the initialized nodes by providing a feature matrix X, thus achieving the best results in the classification of the data.

Since M-GCKE is a keyword extraction algorithm based on GCN, the size of the graph will affect the accuracy of keyword extraction. We use 1000 papers of the Clu_Com dataset as input, with approximately 24,500 nodes, and an average of 8166.7 nodes in one co-occurrence graph. In the experimental result, precision is 8.7%, recall is 9.6%, and F1 value is 9.0%, which has a big gap between that in Table 2. We deem it may be too many nodes in the graph, which causes the co-occurrence frequency of keywords will be higher in the full text, rather than the title and abstract of the paper. If the multi-layer perceptron is not added after the GCN, the experimental results are slightly worse for the same 100 papers, the precision is 15.2%, the recall is 16.8% and the F1 value is 15.9%.

The experimental results in Table 2 show that M-GCKE performs better when the keywords are co-occurring in different papers. When we choose 100 papers in different fields, the co-occurrence number of keywords in different papers is little or no, the experimental results are displayed in Table 3. The two algorithms have very little difference in the accuracy evaluation. We summarize the above two keyword extraction algorithms as follows: S-DWKE is a method to extract keywords for a single paper, which is extensible. When adding a new paper, we do not need to learn the features of all candidate keywords from scratch again, just learn the feature of candidate keywords in new papers. However, it is inefficient because each paper is trained individually to extract features. Therefore, it is suitable for the long-term dynamic detection of hot topics in some mature academic fields. M-GCKE expands the relationship between keywords from a single paper to multiple papers, which can effectively mine more keywords that can reflect topics. Nevertheless, it is not flexible enough because it needs to be retrained the model when the new papers are added. Therefore, it is suitable for the emerging fields in short-term development to predict emerging hot topics.

D. PERFORMANCE FOR HOT TOPIC DETECTION ALGORITHM FOR ACADEMIC BIG DATA

Our ultimate goal is to mine hot topics of a period (one week, one month, one year). Since there is no relevant dataset for academic hotspot research, we manually construct a dataset for evaluating the effectiveness of AHTD. This dataset contains the title, abstract, and keywords of 1000 papers published by the Clustering Computer Journal in 2018. Due to a large number of papers in a year, we choose S-DWKE to extract keywords from them. According to the definition of hot topics in this paper, we extract keywords from the 1000 papers and then get the top 100 hot keywords from

TABLE 4. Predicted 2018 fast-rising trending topics on Clustering Computer journal.

No	Hot Topic
1	resource allocation, image segmentation, energy consumption, energy efficiency, data mining technology, power consumption, resource utilization
2	sensor nodes, feature selection, mobile nodes, task scheduling, malicious nodes, multicast routing, network nodes, mobile node, mobile robot, large data
3	genetic algorithm, particle swarm optimization, medical images, svm model, novel algorithm, prediction model, improved algorithm, image compression, optimization algorithm, new method, optimal design, new algorithm, online shopping, particle swarm algorithm, artificial bee colony algorithm, power optimization, map matching
4	cloud computing, proposed system, image processing, cloud data center, proposed scheme, product innovation, cloud resources, cloud service platform, android system, cloud storage, waiting time, unified framework, live migration
5	big data, data mining, data mining algorithms, data integrity, cloud data, data transmission, sentiment analysis, data set, big data mining, correlation analysis, network lifetime, data analysis, required course, sensor data
6	neural network, target detection, cognitive radio network, parameter estimation
7	wireless sensor networks, mobile ad hoc networks, college students, wireless sensor network, mobile ad hoc network, online social networks, performance analysis, fault diagnosis, feature extraction, face recognition, network function virtualization, network traffic, network performance, supply chain, case study, job scheduling, support vector machine, svm classification
8	fault detection, intrusion detection, extracted features, detection system, association rules, control system, intrusion detection system, intrusion detection systems
9	data clustering, evaluation method, information technology, analysis model, paper studies, control method, performance evaluatio
10	psychological pressure, simulation test, large number

them. Finally, we adopt the K-Means algorithm to cluster hot keywords to get hot topics, which as a final predicted hot topic for this journal in 2018. In this experiment, E in formula (11) is the smallest when the number of hot topics classification k is 10, so we set k to 10 and use the cluster center vector of each topic to measure the topic similarity. Table 4 shows all the predicted hot topics. The experimental results in Table 5 show that our 10 predicted hot topics cover 7 real hot topics, and our predicted hot topics have the highest similarity to real topics.

TABLE 5. Similarity(%) between predicted and real topics.

	T-1	T-2	T-3	T-4	T-5	T-6	T-7
LDA	10	10	30	10	–	–	–
CNM	23.2	42.6	32.7	–	–	–	–
Ours	87.8	60.4	88.3	82.1	85.8	70.3	93.6

Reference [2] proposed a method to automatically find the hot research topic, which is similar in purpose to our model. Therefore, we choose the paper [2] as a comparison algorithm and Latent Dirichlet Allocation (LDA) [33] as a baseline. From the experimental results in Table 6, we can see our method is superior to that of the other two methods for the following reasons. Firstly, the paper [2] uses LDA to extract the keywords directly from the paper, and the extracted keywords are in the form of a single word, including a lot of meaningless keywords (e.g., the, a). However, the real keywords are mostly nouns or combinations of adjectives and nouns. Therefore, we extract candidate keywords from the combination of nouns and adjectives, then propose two different algorithms to generate keywords from candidate keywords, which greatly improves the accuracy of the keywords extraction. In addition, paper [2] uses keywords directly to generate hot topics, and we first get hot keywords from extracted keywords and then use the semantic representation information of hot keywords to generate hot topics, which improves the accuracy of generating hot topics.

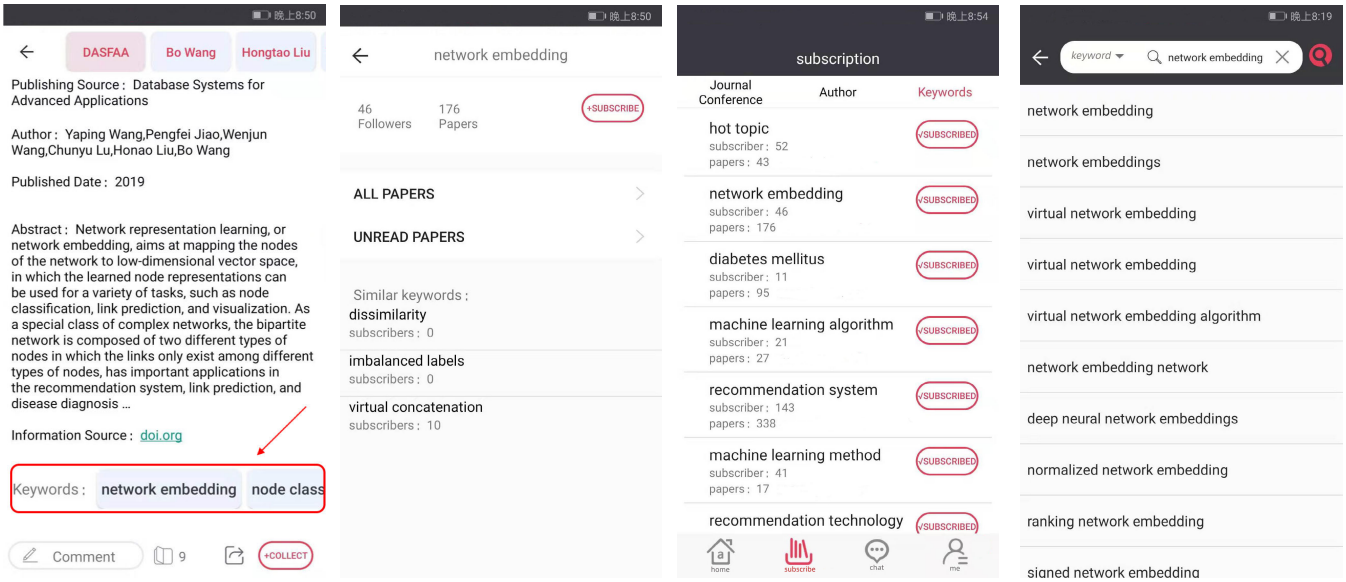
TABLE 6. Performances of different methods.

Algorithm	P(%)	R(%)	F1(%)
LDA	40	40	40
Method in Ref.[2]	65	50	57
Our Model	70	70	70

TABLE 7. Predicted No.4 hot topic and Real No.9 hot topic.

Predicted	Real
cloud computing	cloud computing
cloud data.	big data
cloud data center	cloud cloud services
cloud resources	cloud storage
cloud service platform	cloud data center
cloud storage	—
live migration	—

Table 6 presents the predicted hot topic No.4 and the real hot topic No.9, from which we can observe the accuracy of the predicted hot topics intuitively.



(a) Predicted Keywords of a Paper (b) Keywords Homepage (c) Subscribed Keywords (d) Search Keywords

FIGURE 3. The application scenario of the keyword extraction algorithm. In the (a), users can see the keywords extracted with our keyword extraction algorithm in the paper detail page. On the homepage of the keyword in (b), users can also subscribe to this keyword and see more details information about this keyword, such as how many followers, how many related papers, similar keywords, etc. Words in (c) are all keywords that users have subscribed to, and users can search for keywords in (d). Academic Headline: https://www.acheadline.com.

E. ALGORITHM APPLICATION

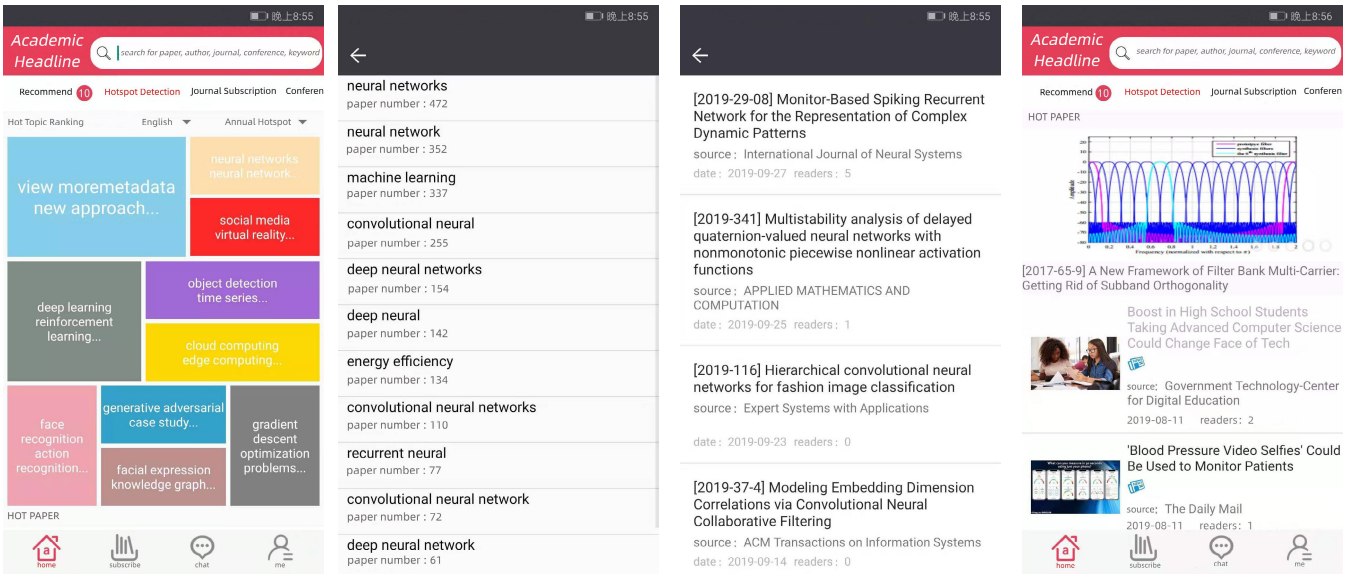
Jilin Province Key Technology Research Project "Research, Development and Application of Rapid Knowledge Sharing System in Big Data and Mobile Internet Era", which has developed "Academic Headline" [3] application. So far, this application contains more than 4.1 million papers, 6000 journals and conferences and more than 300 kinds of academic conferences in the computer field, with more than 6.7 million academic authors and over 1.4 million keywords. As of now, there are more than 7,200 registered users. The paper included in "Academic Headline" application does not contain keywords, but users need to use keyword to search and subscribe to papers. Therefore, it is extremely important to use the keyword extraction algorithm to obtain keywords. In addition, the hot topics detection method can help researchers quickly get the hot topics of the past week, month and year.

The practical usage of the Academic Hot Topic Detection framework (AHTD) corresponds to the framework described in Figure 1, each step of which needs to be implemented. Take the application in the "Academic Headline" app as an example. We divide it into keyword extraction and hot topic recognition according to the framework of AHTD in Figure 1. We deploy the code of the model to the server and set up the timed tasks for keyword extraction and hot topic detection separately. The keyword extraction algorithm is executed once a day to extract keywords of new papers acquired on the same day. The hot topic detection algorithm is executed once a week, twice a month, once a month to update weekly hot topic, monthly hot topic, and annual hot topic. The input is the keywords extracted during the period from the beginning of this week, this month, and this year to the current time point. We divided 12 academic fields according to disciplines and regularly updated their hot topics (e.g., computer science, chemistry, biology, mathematics).

The main duties of the keywords extraction algorithm are reflected in the following aspects. When users browse a paper in the "Academic Headline" app, they can have a quick understanding of this article content through the keywords extracted by our keyword extraction algorithm, which saves users' time, as shown in Figure 3-(a). In addition, the subscription function is customized for the keyword as shown in Figure 3-(b), the user can subscribe to a keyword. When a new paper related to the keyword is published, the system will push it to the user. Here, the user can also see how many individuals subscribed to this keyword, how many papers related to it, all the papers related to it and currently unread papers, and similar keywords calculating by other algorithms. In the subscription center in Figure 3-(c), users can see all the keywords that have been subscribed, and the number of their subscribers and the number of related papers. Finally, users can search for keywords in Figure 3-(d). For example, users search for "network embedding", they will find all tThe he keywords related to it by fuzzy matching.

In the application process of the hot topic detection algorithm, we generate the hot topics of 12 academic fields according to the discipline division field. Taking computer science as an example, we will generate weekly hotspots, monthly hotspots, and annual hotspots separately, and users can select Chinese or English to browse. In Figure 4-(a), the annual hot topics in the computer field are presented, that of all keywords making up the popular topic "neural networks..." are listed in Figure 4-(b). The papers related to its

[3]http://www.acheadline.com/



(a) Annual Hot Topic (b) Keywords under the Hot Topic (c) Hot Papers under "neural networks" (d) Ranking of Hot Papers

FIGURE 4. Hot topic detection results. In the hot topic discovery interface, users can view weekly hotspots, monthly hotspots and annual hotspots in different academic fields in Chinese or English. In (a), it shows the annual hot topics of computer science, and the keywords in the hot topic "neural networks ..." are listed in (b), the papers including the first keyword "neural networks" are shown in (c). Then rank all the papers related to the keywords according to the reading volume, and generate popular paper rankings, as shown in (d).

first keyword "neural networks" can be seen in Figure 4-(c). Moreover, we have generated a hot paper ranking based on the reading volume of papers related to these keywords, as shown in Figure 4-(d).

V. CONCLUSION

In this paper, a general framework to detect hot topics from big academic data is proposed. In this framework, two types of keyword extraction algorithms are proposed to deal with single and multiple papers, respectively, i.e., the S-DWKE and M-GCKE. The former aims at extracting keywords from a single paper based on DeepWalk, and the latter focuses on extracting new keywords from multiple papers in emerging academic areas based on GCN. Then, hot topics can be generated in practical application by keywords extracted by the S-DWKE and M-GCKE accordingly. To validate the effectiveness of the proposed AHTD framework, we conducted a large number of experiments on five datasets, and the results of which demonstrated the advantage of the AHTD. The proposed framework has been applied to the "Academic Headline" app to provide the hot topics for researchers from various academic fields.

The work in this paper on the hot topic detection of academic data is retrospective, which studies how to generate popular topics in the past period. For the future work, we plan to predict academic hotspots in the future, which is of great significance for researchers to grasp the future research direction.

REFERENCES

[1] J. Zhao, H. Wu, F. Deng, W. Bao, W. Tang, L. Fu, and X. Wang, "Maximum value matters: Finding hot topics in scholarly fields," 2017, *arXiv:1710.06637*. [Online]. Available: https://arxiv.org/abs/1710.06637
[2] Z. Zhang and L. Yu, "Academic hot-spot analysis on information system based on the Co-term network," in *Proc. 18th Pacific Asia Conf. Inf. Syst.*, 2014, pp. 109–117.
[3] J. L. Hurtado, A. Agarwal, and X. Zhu, "Topic discovery and future trend forecasting for texts," *J. Big Data*, vol. 3, p. 7, Dec. 2016.
[4] T. Qian, Q. Li, and B. Liu, "Topic formation and development: A core-group evolving process," *World Wide Web*, vol. 17, no. 6, pp. 1343–1373, 2014.
[5] J. Shang, J. Liu, M. Jiang, X. Ren, C. R. Voss, and J. Han, "Automated phrase mining from massive text corpora," *IEEE Trans. Knowl. Data Eng.*, vol. 30, no. 10, pp. 1825–1837, Oct. 2018.
[6] C. Florescu and W. Jin, "Learning feature representations for keyphrase extraction," in *Proc. 32nd AAAI Conf. Artif. Intell.*, 2018, pp. 8077–8078.
[7] S. D. Gollapalli, X. Li, and P. Yang, "Incorporating expert knowledge into keyphrase extraction," in *Proc. 31st AAAI Conf. Artif. Intell.*, 2017, pp. 3180–3187.
[8] A. Hulth, "Improved automatic keyword extraction given more linguistic knowledge," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2003, pp. 216–223.
[9] J. Chen, X. Zhang, Y. Wu, Z. Yan, and Z. Li, "Keyphrase generation with correlation constraints," 2018, *arXiv:1808.07185*. [Online]. Available: https://arxiv.org/abs/1808.07185
[10] Y. Zhang and W. Xiao, "Keyphrase generation based on deep Seq2seq model," *IEEE Access*, vol. 6, pp. 46047–46057, 2018.
[11] Y. Zhang, Y. Fang, and X. Weidong, "Deep keyphrase generation with a convolutional sequence to sequence model," in *Proc. 4th Int. Conf. Syst. Informat.*, Nov. 2017, pp. 1477–1485.
[12] R. Meng, S. Zhao, S. Han, D. He, P. Brusilovsky, and Y. Chi, "Deep keyphrase generation," 2017, *arXiv:1704.06879*. [Online]. Available: https://arxiv.org/abs/1704.06879
[13] K.-Y. Chen, L. Luesukprasert, and S.-C. T. Chou, "Hot topic extraction based on timeline analysis and multidimensional sentence modeling," *IEEE Trans. Knowl. Data Eng.*, vol. 19, no. 8, pp. 1016–1025, Aug. 2007.
[14] Y. Chen, H. Amiri, Z. Li, and T.-S. Chua, "Emerging topic detection for organizations from microblogs," in *Proc. 36th Int. ACM SIGIR Conf. Res. Develop. Inf.*, 2013, pp. 43–52.
[15] W. Xie, F. Zhu, J. Jiang, E.-P. Lim, and K. Wang, "TopicSketch: Real-time bursty topic detection from Twitter," *IEEE Trans. Knowl. Data Eng.*, vol. 28, no. 8, pp. 2216–2229, Aug. 2016.
[16] R. Yu, X. Xie, Y. Li, M. Zhao, X. Dong, M. He, P. Chang, and Z. Wang, "Online hot topic detection based on segmented timeline and aging theory," *Int. J. Hybrid Inf. Technol.*, vol. 9, no. 2, pp. 247–258, 2016.

[17] Y. He, J. Li, M. Zhu, and D. Xu, "Life cycle identification and analysis of microblog hot topics," in *Proc. 10th Int. Conf. Intell. Hum.-Mach. Syst. Cybern.*, Aug. 2018, pp. 156–159.
[18] T. Cao, T. Tran, and T. Luu, "Hot topic detection on newspaper," in *Proc. 9th Int. Symp. Inf. Commun. Technol.*, 2018, pp. 114–121.
[19] X. Yan, J. Guo, Y. Lan, J. Xu, and X. Cheng, "A probabilistic model for bursty topic discovery in microblogs," in *Proc. 29th AAAI Conf. Artif. Intell.*, 2015, pp. 353–359.
[20] X. Yan, J. Guo, Y. Lan, and X. Cheng, "A biterm topic model for short texts," in *Proc. 22nd Int. Conf. World Wide Web*, 2013, pp. 1445–1456.
[21] M. Katsurai, "Bursty research topic detection from scholarly data using dynamic Co-word networks: A preliminary investigation," in *Proc. IEEE 2nd Int. Conf. Big Data Anal.*, Mar. 2017, pp. 115–119.
[22] T. Ma, Y. Zhao, H. Zhou, Y. Tian, A. Al-Dhelaan, and M. Al-Rodhaan, "Natural disaster topic extraction in sina microblogging based on graph analysis," *Expert Syst. Appl.*, vol. 115, pp. 346–355, Jan. 2018.
[23] A. Fang, I. Ounis, C. MacDonald, P. Habel, X. Xiong, and H.-T. Yu, "An effective approach for modelling time features for classifying bursty topics on Twitter," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, 2018, pp. 1547–1550.
[24] M. Hammad, S. R. El-Beltagy, "Towards efficient online topic detection through automated bursty feature detection from arabic Twitter streams," *Procedia Comput. Sci.*, vol. 117, pp. 248–255, Jan. 2017.
[25] C. Florescu and C. Caragea, "A position-biased pagerank algorithm for keyphrase extraction," in *Proc.31st AAAI Conf. Artif. Intell.*, 2017, pp. 4923–4924.
[26] S. N. Kim, O. Medelyan, M.-Y. Kan, and T. Baldwin, "Semeval-2010 task 5: Automatic keyphrase extraction from scientific articles," in *Proc. 5th Int. Workshop Semantic Eval.*, 2010, pp. 21–26.
[27] M. Krapivin, A. Autaeu, and M. Marchese, "Large dataset for keyphrases extraction," Univ. Trento, Trento, Italy, Tech. Rep. DISI-09-055, 2009.
[28] R. Mihalcea and P. Tarau, "Textrank: Bringing order into text," in *Proc. Conf. Empirical Methods Natural Lang. Process.*, 2004, pp. 377–387.
[29] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*. [Online]. Available: https://arxiv.org/abs/1301.3781
[30] B. Perozzi, R. Al-Rfou, and S. Skiena, "Deepwalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2014, pp. 701–710.
[31] T. N. Kipf and M. Welling, "Semi-supervised classification with graph convolutional networks," 2016, *arXiv:1609.02907*. [Online]. Available: https://arxiv.org/abs/1609.02907
[32] N. Shervashidze, P. Schweitzer, and E. J. Leeuwen, "Weisfeiler-lehman graph kernels," *J. Mach. Learn. Res.*, vol. 12, pp. 2539–2561, Sep. 2011.
[33] D. M. Blei, A. Y. Ng, and M. I. Jordan, "Latent Dirichlet allocation," *J. Mach. Learn. Res.*, vol. 3, pp. 993–1022, Jan. 2003.

BEIBEI WANG received the B.Sc. degree from the College of Software, Jilin University, in 2018, where she is currently pursuing the M.S. degree in computer software and theory with the College of Computer Science and Technology. Her main research interest at present stage includes hotspot prediction and event extraction in knowledge graph.

BO YANG received the B.Sc., M.S., and Ph.D. degrees in computer science from Jilin University. He is currently a Professor with the College of Computer Science and Technology, Jilin University. He is also the Director of the Key Laboratory of Symbolic Computation and Knowledge Engineer, Ministry of Education, China. He has published more than 90 articles on international journals including IEEE TPAMI, IEEE TKDE, IEEE TCYB, ACM TWEB, DKE, JAAMAS, and KBS, and international conferences including IJCAI, AAAI, ICDM, WI, PAKDD, and ASONAM. His current research interests are in the areas of data mining, machine learning, knowledge engineering, and complex/social network modeling and analysis. He has served as the Program Chair and SPC for several international conferences. He served as an Associate Editor and peer reviewer for several international journals.

SHUANGSHUANG SHAN received the B.Sc. degree from the Computer Institute, North China University of Technology, in 2016, and the M.S. degree from the Software College, Jilin University, China. Her research interests include social network analysis, knowledge engineering, and data mining.

HECHANG CHEN received the M.S. and Ph.D. degrees from the College of Computer Science and Technology, Jilin University, in 2014 and 2018, respectively. He was enrolled in the University of Illinois at Chicago and Hong Kong Baptist University as a Visiting Ph.D. Student from 2015 to 2016 and from 2017 to 2018, respectively. His current research interests include heterogenous data mining and complex network modeling with applications to computational epidemiology.