The Journal of Supercomputing (2019) 75:2890–2924
https://doi.org/10.1007/s11227-018-2704-z

Automatic content extraction and time-aware topic clustering for large-scale social network on cloud platform

Chunlin Li[1,2,3,4] · Jingpan Bai[2]

Published online: 26 November 2018

Abstract

In recent years, with the increase in users in social network, the social network has had the feature of big data. The large-scale social network has become an indispensable part in people's life. However, the traditional data mining technology cannot suit the large-scale social network. Thus, it is urgent to develop a more suitable mining technology for the large-scale social network. In this section, a crawler model based on semantic analysis and spatial clustering is proposed firstly. Then, the content extraction model based on document object model tree is built to extract the target text information from the links fetched by the proposed crawler model. The similarities between textual information in different regions are computed to choose the important information. Moreover, a two-stage topic clustering model based on time information is presented. The time information is introduced into the similarity computation between two posts or clusters. The single-pass algorithm is improved and applied in different clustering stage to improve the clustering accuracy. Finally, the proposed algorithms are evaluated on Hadoop platform. The Hadoop platform can effectively reduce the computing time and improve the server quality of users in large-scale social network. Meanwhile, the experiments demonstrate that the proposed algorithms are suitable for the data processing in large-scale social network.

Keywords Content extraction · Topic clustering · Network community · Large-scale social network · Cloud platform

✉ Chunlin Li
chunlinli74@163.com

1 Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Land and Resources, Shenzhen, People's Republic of China

2 Department of Computer Science, Wuhan University of Technology, Wuhan 430063, People's Republic of China

3 Key Lab of Guangdong for Utilization of Remote Sensing and Geographical Information System, Guangzhou Institute of Geography, Guangzhou, People's Republic of China

4 Jiangsu Key Laboratory of Meteorological Observation and Information Processing, Collaborative Innovation Center of Atmospheric Environment and Equipment Technology, Nanjing University of Information Science and Technology, Nanjing, People's Republic of China

1 Introduction

In recent years, with the development of medium and network technology, social network has become popular rapidly. In social network, the public reaction for one issue can be reflected by the opinions of users. But, due to the inconsistency and diversity of the expressions, the information in social network is mostly unstructured data. Thus, the data mining technology, such as content extraction, topic clustering, community detection, etc., becomes the urgent requirement on discovering public opinion, public sentiment, etc. Moreover, with the increase in users in social network, the social network has had the feature of big data. The large-scale social network has become an indispensable part in people's life. It not only reflects public opinion and public sentiment, but affects public thought and life style. Sometimes, it posed even a threat to national security [1]. Therefore, we should reasonably utilize the social network to guide public opinion and ensure national security. However, the traditional data mining technology cannot suit the large-scale social network. For example, the time overhead of topic clustering increases with the increase in scale of social network so that large of time overhead cannot satisfy the service requirements. Thus, it is urgent to develop a more suitable mining technology for the large-scale social network so that the public opinion, public sentiment, etc., are easily discovered.

Content extraction is one of the important mining technologies for large-scale social network [2]. It is mainly used to extract the important information from the home page of user and store them into local database. In this paper, for achieving the user links, a crawler model based on semantic analysis and spatial clustering is proposed. Firstly, the vector space model [3] and the term frequency–inverse document frequency model [4] are applied to decide the similarity between contents and target domain. The target domain may be key words, one sentence, etc. Then, the sematic analysis model is created to calculate the weight of each fetched links. The fetched links will be listed in descending order by weight. Finally, the density-based spatial clustering of applications with noise [5] is used to cluster old and new links to guarantee link number that is related target domain.

Moreover, a content extraction model based on document object model tree is built to extract the target text information from the links fetched by the proposed crawler model. Firstly, the characteristics of home page of user based on document object model tree are analyzed. Then, the home page of user is automatically divided into several regions. The similarities between textual information in different regions are computed. Finally, the target text information is extracted and the rest is taken as the noisy text.

Topic clustering also is one of the important mining technologies for large-scale social network. It is mainly used to classify the opinions from users and discovery public opinion. In this paper, the forum is regarded as an example. A two-stage topic clustering model based on time information is presented. Firstly, the time information is introduced into the similarity computation between two posts or clusters by analyzing the characteristics of forum. Then, the single-pass algorithm is applied to cluster the new posts in first stage. Finally, the similarities between the clusters of new posts and the clusters of old posts are computed and the clustering results of the clusters of new posts and the clusters of old posts are achieved in second stage.

The main contributions of this paper are summarized as follows:

- The content extraction model based on document object model tree is built to extract the target text information from the links fetched by the proposed crawler model. The similarities between textual information in different regions are computed to choose the important information.
- A two-stage topic clustering model based on time information is presented. The time information is introduced into the similarity computation between two posts or clusters. The single-pass algorithm is improved to be applied in different clustering stage to improve the clustering accuracy.
- The proposed algorithms are evaluated on Hadoop platform. The Hadoop platform can effectively reduce the computing time and improve the server quality of users in large-scale social network. Meanwhile, the experiments demonstrate that the proposed algorithms are suitable for the data process in large-scale social network.

The rest of this paper is organized as follows: In Sect. 2, several related works are presented. In Sect. 3, the proposed methodologies and the corresponding algorithms are presented. The experimental results are demonstrated in Sect. 4. Lastly, the conclusion of this paper is completed in Sect. 5.

2 Related work

With the development of the social network, a lot of works have been recently conducted to study the content extraction problems and the topic clustering problems. In this section, some brief descriptions about the works are given.

The focused crawler algorithm is the foundation of content extraction. Some researchers have made a lot of works on it. Hassan and Cruz [6] proposed a novel method to process prodigious amounts of information generated by the social network based on an unsupervised and adaptive ontology-learning process. Meanwhile, the most valuable pieces of information were extracted. Bai et al. [7] presented a framework of focused linked data crawler for RDF data stores by considering context graphs. In this framework, the classifiers were trained for detecting and assigning documents to different categories to improve the performance of the framework. Gupta [8] gave a solution to the problems that the crawler retrieved the pages, indexed them and extracted the hyperlinks inside the pages in the automatic publication data gatherer. Vieira et al. [9] discussed the relationship between seed pages and the performance of crawlers. Furthermore, he proposed a new framework of focused crawler for automatically finding the seed pages and improving crawling performance and efficiency.

Almuhareb [10] presented a special focused crawler to detect Arabic poetry resources by improving the Apache Nutch crawler. The special focused crawler implemented its function by using an SVM classifier and a list of Arabic poetry related keywords. Du et al. [11] presented an improved retrieval model, i.e., the semantic similarity vector space model, by combining the vector space model and semantic similarity retrieval model to improve the performance of the corresponding focused crawler. Wei and Li [12] proposed a novel focused crawler by improving genetic algorithm. In the improving genetic algorithm, the topic correlation and importance were simultaneously considered by the fitness function to improve the global search capability of the proposed focused crawler. Boukadi et al. [13] presented a focused crawler for cloud service discovery to guarantees saving the search time and a better exploitation of the provider offerings with a dedicated cloud service description ontology.

It can be known from the works on the focused crawler that the focused crawler needs to be always improved to suit for the special application scenario. There may not be an ideal focused crawler to suit all application scenarios. Therefore, in this paper, a crawler model is proposed to suit for the large-scale social network. The focused crawler model is the foundation of content extraction in the large-scale social network. Furthermore, the content extraction problem will be studied based on the proposed crawler model. The content extraction problem has been discussed by a large amount of scholars.

Pouriyeh et al. [14] formulated RDF data, which was regarded as a very good source of information, for the entity summarization task by combining the topic modeling and the word embedding technique. Luper et al. [15] integrated global positioning systems data, RDF metadata and data mining and proposed an application that is used to analyze the temporal and spatial interaction in an association network environment. Zhang and Ding [16] introduced the ontology into content extraction to improve precision rate, recall rate and practical performance of content extraction. The content extraction is implemented based on the ontology extraction rules. Fagin et al. [17] built a framework to avoid the inconsistencies in IE based on the database theory. The prioritized repairs strategy was adopted to incorporate priorities among conflicting facts for improving the performance of content extraction. Velasco et al. [18] analyzed the architectural pattern descriptions in terms of specific quality attributes by combining knowledge representation and information extraction. Gao et al. [19] presented a content extraction strategy based on time–frequency transformation to achieve the high-quality contents from social network. Moreover, a new automatic query-reply mechanism in social network was proposed based on the proposed content extraction strategy.

The content extraction is used to extract the important information from social network. Then, the extracted information is stored into local database for data mining, topic clustering, community detection, etc. In this paper, the topic clustering method is discussed based on the proposed content extraction method. Likewise, there are also amount of literatures on topic clustering.

Mehdi et al. [20] proposed a knowledge-based topic model by integrating an ontology as a knowledge base into the statistical topic models in a principled way. Furthermore, a labeling method is defined to enhance the accuracy of topic model. Seyedamin et al. [21] proposed a probabilistic topic model by creating a single framework, which included prior knowledge and statistical learning techniques, to provide more reliable and representative summaries for entities. Yeh et al. [22] presented a dynamic topic clustering model by considering the proportions of verbs and nouns to capture the sequence of two adjacent topics in spoken content. Lin et al. [23] proposed a term-based consensus clustering topic detection framework, which was an unsupervised methodology, to detect distinct topics from within short text systems collections. Meanwhile, the K-means-based consensus clustering method was used to address short text systems clustering, because this method had low computational complexity and robust clustering performance.

Chakraborti and Dey [24] studied the effectiveness of K-means text clustering algorithm that was adopted on multiple levels, in a top-down, divide-and-conquer fashion, on competitor intelligence corpus. Meanwhile, the capability of multi-level K-means clustering technique, which was used to determine the optimal number of clusters as part of clustering process, was demonstrated. Hashimoto et al. [25] induced an informative representation of studies into the topic detection method to improve the performance of the underlying active learner. In addition, the semantic similarities between documents are calculated by a neural network-based vector space model. Zhang et al. [26] presented a hybrid relations analysis approach to discovery the relations and co-occurrence relations for topic clustering so that the topic clustering is more effective and accurate. Nguyen [27] innovatively combined the frequency information and the international patent classification to address the problem of semantic information on word categorization for topic clustering.

In this paper, a topic clustering algorithm that is more suitable to large-scale social network is proposed, which is different with previous works. The single-pass algorithm is improved by analyzing the feature of forum. Meanwhile, the time information of posts is introduced into the similarity computation between two posts so that the topic clustering is more effective and accurate.

3 Proposed methodology

In this section, firstly, a crawler model based on semantic analysis and spatial clustering is proposed to improve efficiency and accuracy of fetched data. Furthermore, a content extraction model based on document object model tree is built to extract the target text information from the links fetched by the proposed crawler model. Then, the extracted information is stored into local database for data mining, topic clustering, community detection, etc. Moreover, based on the proposed content extraction, a topic clustering method is discussed with time information method to improve the accuracy of the information retrieval.

3.1 Crawler model based on sematic analysis and spatial clustering

The crawler model is a key component of search engine system. A crawler model is mainly used to fetch data from social network and provide data for search engines. The data mainly includes textual and link information. However, crawler models face some challenges. For example, how to fetch more targeted data for the specific users is the key technique of a crawler method. In this section, a crawler model based on sematic analysis and spatial clustering is proposed to improve efficiency and accuracy of fetched data. Firstly, the vector space model [3] and the term frequency–inverse document frequency model [4] are applied to decide the similarity between contents and target domain. The target domain may be key words, one sentence, etc. Then, the sematic analysis model is created to calculate the weight of each fetched links. The fetched links will be listed in descending order by weight. Finally, the density-based spatial clustering of applications with noise [5] is used to cluster old and new links to guarantee link number that is related target domain.

Let $C = (c_1, c_2, \ldots, c_n)$ denote n-dimension vector that is composed from keyword entries of the target domain. A n-dimension vector that consists of keyword entries that wait for analysis is taken by $C_i = (c_{i1}, c_{i2}, \ldots, c_{in})$. φ_k and φ_{ik} are the weight of c_k and c_{ik} respectively, which can be calculated by term frequency–inverse document frequency. φ and φ_i are the weight vector of C and C_i, respectively. Then, the similarity between C and C_i is of the form: $Sim(C, C_i) = \cos(\varphi C, \varphi_i C_i)$. Given a threshold σ. If $Sim(C_i, C_j) > \sigma$, the link that waits for analysis is related to the target domain. If $Sim(C_i, C_j) \leq \sigma$, the link that waits for analysis is not related to the target domain.

When a link is obtained, several new links can be achieved by source code of that link. In this case, that link can be taken as father link and the new links should be regarded as son links. The relationship between target domain and son link not only is influenced by the father links, but is affected by the contents of these two entries.

Let $inheritedSim$ represent the inherited similarity of son link from father link. If father link is related to target domain, $inheritedSim = \alpha Sim(fartherlink, targetdomain)$, where $\alpha(\alpha \in [0, 1])$ is the damping factor form father link to son link. The similarity between target domain and father link is defined by $Sim(fartherlink, targetdomain)$. $inheritedSim = \alpha Gfartherlink$ if father link is not related to target domain, where $Gfartherlink$ is taken as the genetic factor, which is predefined constant.

Next, the external similarity between son link and target domain will be computed by anchor texts and contents of son link. Let $anchortextSim = Sim(anchortext, targetdomain)$ represent the similarity between the target domain and the anchor text of son link. The similarity between the target domain and the content of son link is defined by $contentSim = Sim(content, targetdomain)$. Then, the external similarity between son link and target domain is of the form $externalSim = \beta * anchotextSim + (1 - \beta) * contentSim$, where β is the weight of $anchotextSim$.

Therefore, the similarity between son link and target domain is of the form

$$Sim(sonlink, targetdomain) = \gamma * inheritedSim + (1 - \gamma) * externalSim, \tag{1}$$

where γ is the weight of $inheritedSim$, $\gamma \in [0, 1]$.

Density-based spatial clustering of applications with noise is a method to analyze all fetched links, including both old and new links, for clustering. In this method, the number of user links covered by area of radius r should be greater or equal to the given threshold so that there are enough links to be fetched for target domain.

Let $L = (l_1, l_2, \ldots, l_n)$ be the link vector, where l_i represents ith keyword in the link. For clustering the old links based on the text structure and content similarity in link text, density-based spatial clustering of applications with noise is applied. The generated link clusters by density-based spatial clustering of applications with noise is $Linkcluster = (cluster_1, cluster_2, \ldots, cluster_n)$. $clusternumber = (cn_1, cn_2, \ldots, cn_n)$ is taken as the link number of each cluster. Then, the inherited similarity between new link and old links can be expressed by $inheritedSim(newlink, oldlinks) = \sum_{i=1}^{n} Sim(cluster_i, L)cn_i$. The external similarity between new link and keyword I of target domain is $externalSim(newlink, oldlinks) = \sum_{i=1}^{n} Sim(l_i, I)$. So, the similarity score is

$$Simlarityscore = \eta inheritedSim(newlink, oldlinks) + (1 - \eta)externalSim(newlink, oldlinks), \quad (2)$$

where η is the weight of $inheritedSim(newlink, oldlinks)$, $\eta \in [0, 1]$. The new links will be listed in descending order by $Simlarityscore$.

Algorithm 1 shows the pseudo-code of the proposed crawler model based on sematic analysis and spatial clustering. Firstly, all new links in the user home pages whose link have been achieved should be obtained according to the achieved link list (algorithm 1 line 1). Then, the similarity between each new link and target domain is computed (algorithm 1 line 2–10). According to given threshold, the new link is decided the relationship between new link and target domain (algorithm 1 line 11–13). Finally, the new link that is related to target domain is assigned in the link cluster of target domain and the new links will be listed in descending order by similarity score (algorithm 1 line 14–18).

Algorithm 1：The proposed crawler algorithm based on sematic analysis and spatial clustering

```
Input: The link list that has been achieved $Linklist$
Output: The new link list $Newlinklist$, in which all new links are listed in descending order by similarity score
1  Obtain all new links $Newlinks$ in the home pages of users whose links have been achieved
2  for each $newlink_i \in Newlinks$ do
3    if $newlink_i \in Linklist$
4      continue;
5    endif
6    Calculate $inheritedSim$ // The inherited similarity of son link from father link
7    Calculate $anchortextSim$ // The similarity between the target domain and the anchor text of son link.
8    Calculate $contentSim$ // The similarity between the target domain and the content of son link
9    Calculate $externalSim$ // The external similarity between son link and target domain
10   Calculate $Sim(newlink_i, targetdomain)$ // The similarity between $newlink_i$ and target domain
11   if $Sim(newlink_i, targetdomain) > \delta$
12     $newlink_i$ is regarded to be related to target domain
13   end if
14   Calculate $inheritedSim(newlink, oldlinks)$ // The inherited similarity between new link and old links
15   Calculate $externalSim(newlink, oldlinks) = \sum_{i=1}^{n} Sim(l_i, I)$ // The external similarity between new link and target domain
16   Calculate $Simlarityscore(newlink_i)$
17   $Linklist \leftarrow newlink_i$ // The new links will be listed in descending order by $Simlarityscore(newlink_i)$
18 end for
```

The time complexity of algorithm 1 mainly consists of the time complexity of similarity judgement of link texts and the time complexity of similarity judgement of home pages of users. The time complexity of similarity judgement of home pages of users is $O(n \log C)$, where n is the number of keywords in local database and C is taken as the number of keywords in home pages of users. The time complexity of similarity judgement of link texts is $O(n \log C) + O(m^2)$, where n is denoted as the number of link clusters. C is the vector dimension. m is defined as the number of new links. Therefore, the time complexity of algorithm 1 is $O\left(m^2\right)$.

3.2 Content extraction model based on document object model tree

Document object model [28] is a special programming interface of standard API and is mainly used to process and analyze semi-structured data, such as HTML, XML, etc. Document object model takes the information to be processed as the data structures that have the characters of tree structure and provides the access model for overall analysis object. Currently, most of home pages of users include not only page title, dominant text blocks, comments, other items that are related to page topic, but some introduction information of the user home page, sponsored links, other items that are not related to page topic. The textual information that is not related to page topic, which is also named as noisy text, may be included in the blocks that are related to page topic or in the blocks that are not related to page topic. So, we need to extract and store the textual information that is related to page topic, which is named as target text, into local database for information process or analysis. However, how to efficiently extract the target text becomes a challenge. In this section, a content extraction model based on document object model tree is built to extract the target text information of user home page. Firstly, the characteristics of user home page based on document object model tree are analyzed. Then, the user home page is automatically divided into several regions. The similarities between textual information in different regions are computed. Finally, the target text information is extracted and the rest is taken as the noisy text.

Based on the analysis of document object model tree, some characteristics of user home page are summarized as follows:

- The label portfolios of target text are similar.
- The label portfolios of noisy text are similar.
- The target text is usually stored into the leaf node of document object model tree.

In general, one textual block of user home page can be regarded as noisy text if it satisfies any one of following conditions, which is named initial conditions.

- The information that is not related to the page topic.
- The template information that is provided by the social network provider.
- The advertisement information.

However, these conditions can not find some special noisy text that is mixed into target text by the developer. So, the text similarity is used to discover the noisy text in the round. The initial conditions are used to extract the keyword for each textual block in the user home page. The word vector is denoted by $V = \{v_1, v_2, \ldots, v_n\}$. $W = \{w_1, w_2, \ldots, w_n\}$ is the weight vector of word vector. Then, we can calculate the similarity between each word and target domain. The similarity vector for the word vector is taken as $S = \{s_1, s_2, \ldots, s_n\}$. Let $S_{\max} = \max\{s_1, s_2, \ldots, s_n\}$. Then, the textual block with $S_{\max}$ can be regarded as the body text of user home page. But other textual blocks may include some information that is related to page topic and is not easily detected. Thus, we need to calculate the score of each textual block. The score of textual blocks that has similarity s_m is computed by

$$score_m = (1-\eta)\frac{s_m}{S_{\max}} + \eta s_m. \tag{3}$$

Give a threshold, if the score is less that the threshold, the corresponding textual block is regarded as noisy text. Otherwise, the corresponding textual block is taken as the block that includes target text.

Algorithm 2 shows the pseudo-code of the proposed content extraction algorithm based on document object model tree. Firstly, the document object model tree is initialized (algorithm 2 line 1). Each leaf node in the document object model tree is defined by a key-value pair $\langle tagpath, content\rangle$, where $tagpath$ and $content$ denote tag path and content in the corresponding leaf node, respectively. Then, all script or style blocks in user home page should be removed due to uselessness of them (algorithm 2 line 4). Furthermore, all textual blocks that include target text are extracted (algorithm 2 line 7–10). Finally, the height of document object model tree is reduced one unit and the tag is removed (algorithm 2 line 13–15). When the value of height of document object model tree is 1, all textual information that is related to page topic is extracted.

Algorithm 2：Content extraction algorithm based on document object model tree

```
Input：Source code of home pages of users
Output：The textual information that is related to page topic
1   Initialize target text $\langle tagpath, content\rangle$ and document object model tree
2   do until the value of height of document object model tree is 1
3     for $n \in N$ // $N$ is the set of leaf nodes in document object model tree
4       if $textblock_n$ is script or style block // $textblock_n$ is the text block of leaf node $n$
5         continue;
6       endif
7       Calculate word vector $V$ and similarity and similarity vector for the word vector $S$
8       Calculate $score_n$ // The score of textual blocks for leaf node $n$
9       if $score_n \geq \lambda$
10        $\langle tagpath, content\rangle \leftarrow leaf\ node\ n$
11      end if
12    end for
13    $h--$
14  end for
15  Remove the tag path in $\langle tagpath, content\rangle$
```

The time complexity of algorithm 2 is $O(hlogN)$, where N is the number of leaf nodes in the document object model tree and h denotes the height of the document object model tree. Because algorithm 2 will run on Hadoop platform, the computing time of algorithm 2 should be divided by the number of computing nodes in the Hadoop cluster. In other words, the time complexity of algorithm 2 is far less than $O(hlogN)$ in practice.

3.3 Topic clustering model based on time information

In general, the Internet forum or online community is formed by different and inter-linked communities. In Internet forum or online community, users express their opinions by publishing topics. Usually, a topic can be described by several keywords, i.e., these keywords can form a sequence to represent the topic. Then, other users can participate and discuss the topics based on their interests. However, the complicated and overloaded Internet information brings challenges on how to rapidly and precisely achieve interest topics. Thus, the topic clustering model based on time information is presented to provide convenience for the information retrieval. In this section, we use forum as an example. The characteristics of forum are analysis. Then, all posts in the forum are clustered by improved single-pass (ISP) algorithm so that the posts are classified and the topics are easily found.

The vector space model is used to express the contents in some post, in which each dimension represents one keyword. The term frequency–inverse document frequency model is used to calculate the weight of each keyword. Thus, post q can be defined by $post_q = \{d_{q1} : e_{q1}; d_{q2} : e_{q2}; \cdots ; d_{qn} : e_{qn}\}$, where d_p denotes pth keyword. e_p is the weight of pth keyword. In this case, the matching problem of posts can be converted into the matching problem between vectors. The similarity between post q and r is $Sim(post_q, post_r) = \cos(post_q, post_r)$.

Single-pass (SP) is mainly used to cluster the posts. Firstly, a threshold should be given. The first post is regarded as initial clustering post. Then, a new post is chosen in chronological order. The similarity between new post and some cluster will be achieved. If that similarity is more than the given threshold, the new post will be assigned into this cluster. Otherwise, a new cluster will be created with only this new post.

However, in SP algorithm, a new post need be calculated by using the similarity with each cluster, which leads to numerous of time overhead. For instance, there exist 10 thousand new posts and 1 million clusters. The number of computing times is at least 10 billion. For reducing computing times and improve computing speed, we can make the cluster analysis for 10 thousand new posts to form new clusters. Then, each of new clusters will be computed in similarity with each of 1 million clusters. This method will significantly reduce computing time overhead and improve user experience. For example, the 10 thousand clusters can be classified as 700 new clusters and the number of computing times is 10 thousand. The number of computing times between 700 new clusters and 1 million clusters is at most 700 million. Thus, the overall number of computing times between 10 thousand new posts and 1 million clusters is at most 700 million and 10 thousand. This cluster process is the main idea of ISP. The number of computing times in ISP algorithm can achieve 90% reduction compared with SP algorithm.

A post can be formulated as follows:

$$post_q = (ID_q, title_q, content_q, createtime_q, url_q, author_q, reposters_q, recontents_q), \tag{4}$$

where ID_q, $title_q$, $content_q$, $createtime_q$, url_q and $author_q$ denotes the ID, title, content, creation time, URL, author about the post q, respectively. $reposters_q$ is taken as the list of users that response the post. $recontents_q$ is denoted as the list of response contents.

A topic can be formulated as follows:

$$topic_z = (ID_z, keywords_z, createtime_z, lastmodifytime_z, postID_z), \tag{5}$$

where ID_z, $keyword_z$, $createtime_z$ and $lastmodifytime_z$ are defined as ID, keyword, creation time, last modification time in the topic, respectively. $postID_z$ is taken as the list of posts in the topic.

It is noteworthy that the keywords in post title are more important than ones in other location of the post. The keywords in post title provide more information about the content of the post. Therefore, we should give more weight for the keywords in post title. Let y_{qx} be the weight of keyword x in post q. y_{qx} takes the form of

$$y_{qx} = \frac{tf_{qx} e^{\frac{g_x}{g}}}{G\sqrt{\sum_{x=1}^{g} tf_{qx}^2}} \times local(x), \tag{6}$$

$$local(x) = \begin{cases} \chi > 1, & \text{keword } x \text{ in post title} \\ 1, & \text{otherwise} \end{cases}, \tag{7}$$

where G is the number of posts. g_x is taken as the number of posts that include keyword x. g represents the number of keywords in post q. tf_{xa} is the frequency of keyword x in post q.

Investigations shows a hot topic usually experience the incubation phase, growth phase, maturity phase and decline phase. The analysis about posts shows that the posts that are similar in times may describe same topic. Thus, the similarity computation between posts or topics should consider the published time of posts.

Let the release time of first post for some topic be t_{topic}^{first}. t_{topic}^{last} is the release time of last post for the topic. t_{post} denotes the release time of some post. Then, the release time of the topic is

$$t_{topic} = \sqrt{\frac{\left(t_{topic}^{first}\right)^2 + \left(t_{topic}^{last}\right)^2}{2}}. \tag{8}$$

The release time interval between the post and the topic takes the form

$$timeinterval(t_{post}, t_{topic}) = t_{post} - \sqrt{\frac{\left(t_{topic}^{first}\right)^2 + \left(t_{topic}^{last}\right)^2}{2}}. \tag{9}$$

Then, the similarity between post q and r is the form

$$Sim(d_q, d_r) = \varepsilon \times \frac{\sum_{k=1}^{n} y_{qk} \times y_{2x}}{\sqrt{\left(\sum_{k=1}^{n} y_{qk}^2\right)\left(\sum_{k=1}^{n} y_{qk}^2\right)}} + (1 - \varepsilon) \times timeinterval(t_q, t_{topic}), \tag{10}$$

where t_{topic} denotes the release time of the topic that includes post r. If the similarity between post q and r is more than a given threshold θ, the two posts belong to the same topic. Otherwise, post q and r are belong to different topics, respectively.

When a post is clustered into a topic, the vector center will be changed. In our paper, the vector center can be calculated by

$$Vectorcenter(topic_z) = \frac{1}{Q} \sum_{q=1}^{Q} y_{zq}, \tag{11}$$

where $topic_z$ is the weight of topic z. Q denotes the number of posts in the cluster for the topic. The vector center of topic z is defined by $Vectorcenter(topic_z)$. When a post or a topic is clustered into topic z. The update process of vector center is formulated as follows:

$$Vectorcenter(topic_z) = \frac{1}{Q+1}(QVectorcenter(topic_z) + y^{'}), \tag{12}$$

where $y^{'}$ is the weight of the post or the topic is clustered into topic z.

The topic clustering algorithm based on time information mainly includes two stages. In first stage, the new posts are clustered and the clusters of new posts are achieved (algorithm 3). In second stage, the similarities between the clusters of new posts and the clusters of old posts are computed and the clustering results of the clusters of new posts and the clusters of old posts are achieved (algorithm 4). Algorithm 3 and algorithm 4 depict the clustering process of two stages, respectively.

Algorithm 3 shows the pseudo-code of the clustering algorithm for the new posts. Firstly, the titles and contents in posts are formulated by vector space model (algorithm 3 line 2–5). Then, if the post is the first post, the post is regarded as the first topic. Otherwise, the similarity between the post and existed post are calculated (algorithm 3 line 7–11). If that similarity is more than the given threshold, the new post will be clustered into the cluster of the existed posts and the vector center of the corresponding topic will be updated (algorithm 3 line 12–14). Otherwise, a new topic will be created by the new post (algorithm 3 line 16). Finally, the clustering results of the new posts are achieved.

Algorithm 3: The clustering algorithm for the new posts

```
Input: The post list: $List\langle post\rangle$ . The threshold value: $thresholdValue$ .
Output: The topic list of new posts: $List\langle mid\_topic\rangle$ .
1 Begin
2 for (int $i$=0; $i < List\langle poster\rangle.size$ ; $i++$ )
6   for ( $k$ =0; $k < List\langle mid\_topic\rangle.size$; $k++$ ) // Travers all existed topics
8      $tempValue = Co\sin\_dis(p, List\langle mid\_topic\rangle[i])$
9      if ( $tempValue > maxValue$ )
10        $maxValue = tempValue$
11        $maxIndex = k$
12     if ( $maxValue$ > $thresholdValue$ )
13        add2Topic ( $List < mid\_topic > [maxIndex]$ ) // Join temporary topic $mid\_topi$
14        Update the vector center of temporary topic
15     else
16        Create topic ( $List < mid\_topic > [k+1]$ ) // Create a new topic
```

Algorithm 4 shows the pseudo-code of the clustering algorithm for clusters of new posts and the existed clusters. Firstly, the similarity between temporary topic and each existed topic is computed (algorithm 4 line 5–7). Then, if the similarity is more that the given threshold, these two topics will be merged into one topic (algorithm 4 line 11–12). Otherwise, a new topic will be created by the temporary topic (algorithm 4 line 8–10). Finally, the cluster result for the clusters of new posts and the existed clusters is achieved.

Algorithm 4: The clustering algorithm for clusters of new posts and the existed clusters

```
Input: The topic list of new posts: $List\langle mid\_topic\rangle$ . The threshold value: $thresholdValue$ .
Output: The updated topic list with new posts $List < last\_topic >$
1 Begin
2 for (int $i$=0; $i < List < mid\_post > .size$ ; $i++$){
3    int $k = 0$, $maxIndex$ =0
4    double $tempValue$ , $maxValue$
5    for ( $k$ =0; $k < List\langle mid\_topic\rangle.size$ ; $k++$ ) // Travers all existed topics
6       for ( $j = 0$ ; $j < List < lastday\_topic > .size$ ; $j++$ ){
7          $tempValue = Cosin\_dis(list < last\_topic > [k], list < mid\_topic > [j])$
8          if ( $tempValue > maxValue$ )
9             $maxValue = tempValue$
10            $maxIndex = k$ }
11         if ( $maxValue > thresholdValue$ ){
12           Merge two topics
13         Update the vector center of topic
14    }
15         else
16           add2Topic ( $List < mid\_topic > [j+1]$ ) // Join a topic
17 }
18 End
```

The time complexity of SP algorithm is $O(mkt)$, where m, k and t denote the number of posts, the number of clusters and iteration times. However, in our paper, the Hadoop platform is used to parallelly run the proposed algorithm. The time complexity of SP algorithm is $O((mkt)/nl)$, where n is taken as the number of nodes in Hadoop platform, and l represents the number of tasks run simultaneously on one node. For ISP algorithm, SP algorithm is applied in each stage. Therefore, the time complexity of ISP algorithm is $O((mkt)/nl)$.

4 Experimental results

In this paper, we mainly proposed two algorithms: the content extraction (CE) algorithm and topic clustering (TC) algorithm. In CE algorithm, the proposed crawler (PC) algorithm is the foundation. Furthermore, the forum is taken as example to illustrate TC algorithm based on contents extracted by CE algorithm. TC algorithm mainly includes two stages. In first stage, the new posts are clustered and the clusters of new posts are achieved. In second stage, the similarities between the clusters of new posts and the clusters of old posts are computed and the clustering results of the clusters of new posts and the clusters of old posts are achieved. In this section, extensive experiments are conducted to value the performance of the proposed algorithms.

4.1 Experimental environment

In this section, the proposed algorithms will be experimentally verified. The experimental environment consists of 3 computers, where one is taken as the master and the others are taken as the slavers. Each computer is composed of Intel 2.5 GHz Core i5 CPU running at 4G RAM and 1G HDD. The operating system in each computer is CentOS7.0. The version of the java development kit is JDK 1.7 and the version of the NoSQL development is Hadoop 0.20.2 and Hbase0.94.12. The fundamental crawler framework is Heritrix3.10.

4.2 Test case

In the experiments for PC algorithm, military is taken as the keyword. 10 links that are related to military and 15 links that are not related to military are chosen as text data, i.e., the initial number of links is 25. The number of home pages for users that are used for experiments is 4000. The download links will be clustered every 100 links. In the experiments of content extraction, 101 home pages of users that are downloaded by Maze (http://e.pku.edu.cn/) and 100 home pages for users that are related to Burma Civil War and are downloaded by Google are regarded as the test data. The two datasets are named as dataset I and dataset II, successively. In the experiments for the topic clustering algorithm, all posts in the End of the World Forum are extracted by CE algorithm and are taken as the test data.

4.3 Benchmark algorithm

For evaluating the performance of PC algorithm, best first search [29] (BFS) algorithm and shark search [30] (SS) algorithm are taken as the compared algorithms. In SS algorithm and BFS algorithm, only the relationship between link and target domain is considered, and the correlation between links is neglected. However, PC algorithm considers not only the relationship between link and target domain, but both the correlation between links and the correlation between old link and new link. Therefore, it is reasonable that SS algorithm and BFS algorithm are regarded as the compared algorithms. For evaluating the performance of CE algorithm, five content extraction algorithms proposed by five participating teams in symposium of search engine and web mining (SEWM 2008) (http://net.pku.edu.cn/~sewm/) are taken as the compared algorithm.

For evaluating the performance of TC algorithm, the SP algorithm is taken as the compared algorithm to evaluate the performance of ISP algorithm. SP is mainly used to cluster the posts. Firstly, a threshold should be given. The first post is regarded as initial clustering post. Then, a new post is chosen in chronological order. The similarity between new post and some cluster will be achieved. If that similarity is more than the given threshold, the new post will be assigned into this cluster. Otherwise, a new cluster will be created with only this new post. However, ISP algorithm divides the clustering process of SP algorithm into two stages. In first stages, the new posts are clustered and the clusters of new posts are achieved. In second stage, the similarities between the clusters of new posts and the clusters of old posts are computed and the clustering results of the clusters of new posts and the clusters of old posts are achieved. The time overhead of topic clustering by ISP algorithm is far less than that by SP [31] algorithm, which is illustrated by a sample in paragraph 4 in Sect. 3.3. Therefore, it is reasonable that SP algorithm is regarded as the compared algorithms.

Furthermore, k-means [31] (KM) algorithm and SP algorithm with single machine environment are regarded as the compared algorithms to value the performance of ISP algorithm with parallel operation. In this paper, the application of Hadoop platform in topic clustering can significantly reduce the time overhead of clustering process. KM algorithm is the traditional clustering algorithm. The clustering process of KM algorithm is analogous to that of SP algorithm. But the number of clusters need to be given by experts or experienced people. The clustering results by KM algorithm are subjective, which leads to lower accuracy. SP algorithm is the foundation of ISP algorithm. KM algorithm and SP algorithm with single machine environment are regarded as the compared algorithms to value the performance of ISP algorithm with parallel operation, which can illustrate not only improved performance of ISP algorithm, but the performance of parallel computation. Thus, it is reasonable that KM algorithm and SP algorithm are regarded as the compared algorithms.

4.4 Evaluation metrics

In the experiments for PC algorithm, the precision and recall are chosen as the evaluation metrics. Precision is the ratio of the number of extracted links that are related to target topic (NELRTT) to the number of all extracted links (NAEL), which is denoted by $P_{crawler}$, i.e.,

$$P_{crawler} = \frac{NELRTT}{NAEL}. \tag{13}$$

Recall is computed by the percentage of the number of extracted links that are related to target topic (NELRTT) among the number of all links that are related to target topic (NALRTT), which is defined by $R_{crawler}$, i.e.,

$$R_{crawler} = \frac{NELRTT}{NALRTT}. \tag{14}$$

In the experiments for CE algorithm, also the precision and recall are chosen as the evaluation metrics. Bur, here, precision is the ratio of the number of right extraction information (NREI) to the number of all extraction information (NAEI), which is denoted by P_{CE}, i.e.,

$$P_{CE} = \frac{NREI}{NAEI}. \tag{15}$$

Recall is computed by the percentage of the number of right extraction information (NREI) among the number of all information that should be extracted (NAIE), which is defined by R_{CE}, i.e.,

$$P_{CE} = \frac{NREI}{NAIE}. \tag{16}$$

Furthermore, $F-measure$ is regarded as the evaluation metric, which is expressed by F_{CE}.

$$F_{CE} = \frac{(\beta^2+1)P_{CE}R_{CE}}{\beta^2 P_{CE} + R_{CE}}, \tag{17}$$

where β is a predefined value. It is usually used to balance precision and recall. In general, β is set to 1, 1/2 or 2.

In the experiments for TC algorithm, there are the average false positive rate, the average false negative rate and the average cost of error execution in a system. The average false positive rate is denoted by $AFPR$, which takes the form of

$$AFPR = \frac{\sum_{z=1}^{\Delta} AFPR_z}{\Delta}, \tag{18}$$

where Δ is taken as the number of topics. $AFPR_q$ denotes the false positive rate of topic z, which is of the form

$$AFPR_q = \frac{NUP_z}{NAP_z}, \tag{19}$$

where NUP_z is defined as the number of undiscovered posts that are related to topic z. NAP_z is taken as the number of all posts on topic z.

The average false negative rate is defined by $AFNR$, which is of the form

$$AFNR = \frac{\sum_{z=1}^{\Delta} AFNR_z}{\Delta}, \tag{20}$$

where Δ is taken as the number of topics. $AFNR_z$ is defined as the false negative rate of topic z, which takes the form of

$$AFNR_z = \frac{NNP_z}{NANP_z}, \tag{21}$$

where NNP_z denotes the number of posts that are not related to topic z but in the discovered posts. $NANP_z$ is defined as the number of all posts that are not related topic z.

Let the average cost of error execution in a system be $ACEE$, which is of the form

$$ACEE = CEE_{AFPR} \times AFPR \times Pr_{topic} + CEE_{AFNR} \times AFNR \times (1 - Pr_{topic}), \tag{22}$$

where CEE_{AFPR} and CEE_{AFNR} is the error execution cost of false positive and false negative, respectively. Pr_{topic} denotes the prior probability of generated topic. In actual, the average cost of error execution in a system need be normalized. The normalization of average cost of error execution is denoted by $\overline{ACEE}$.

$$\overline{ACEE} = \frac{1}{ACEE}. \tag{23}$$

$\overline{ACEE}$ is an important indicator to evaluate the performance of clustering algorithm. The performance of clustering algorithm varies inversely with the value of $\overline{ACEE}$. In other words, as the performance of clustering algorithm becomes better, the value of $\overline{ACEE}$ becomes bigger.

4.5 The impact analysis of parameters

4.5.1 The impact analysis of parameter γ

Parameter γ is one of the important parameters in PC algorithm. It is mainly used to balance the effect of inherited similarity and external similarity correlations between target domain and son link. In the experiment for the impact analysis of parameter γ, the control various method is adopted to achieve the most suitable value for our proposed algorithm. Other value of parameters is 0 or 1. The value of parameter γ ranges from 0 to 1. Parameter γ is determined by $P_{crawler}$ and $R_{crawler}$. $P_{crawler}$ and $R_{crawler}$ are computed based on different value of parameter γ, respectively.

Figure 1 depicts the value of $R_{crawler}$ by varying the value of parameter γ. The value of other parameter is set to 0, and the number of extracted home pages for users is 4000. In this case, with the increase in the value of parameter γ, the value of $R_{crawler}$ increases. When $\gamma = 0.7$, $R_{crawler}$ achieves highest value 0.82. Therefore, the value of parameter γ should be set to 0.7 according to $R_{crawler}$, which will be better suitable for PC algorithm.

Figure 2 illustrates the value of $P_{crawler}$ by varying the value of parameter γ. The value of other parameter is set to 0, and the number of extracted home pages for users is 4000. In this case, with the increase in the value of parameter γ, the value of $P_{crawler}$ ranges between 0.65 and 0.75. When $\gamma = 0.8$, $P_{crawler}$ achieves highest value 0.75. But, the value of $P_{crawler}$ fluctuates around 0.7, which implies that the value of parameter γ has little impact on $P_{crawler}$. Thus, the value of parameter γ can be also set to 0.7 according to $P_{crawler}$. In this case, the value of parameter γ is also suitable for PC algorithm .

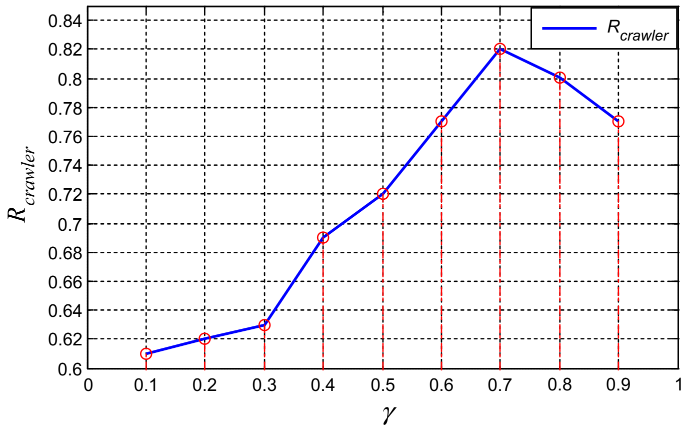


Fig. 1 Recall with different value of parameter γ

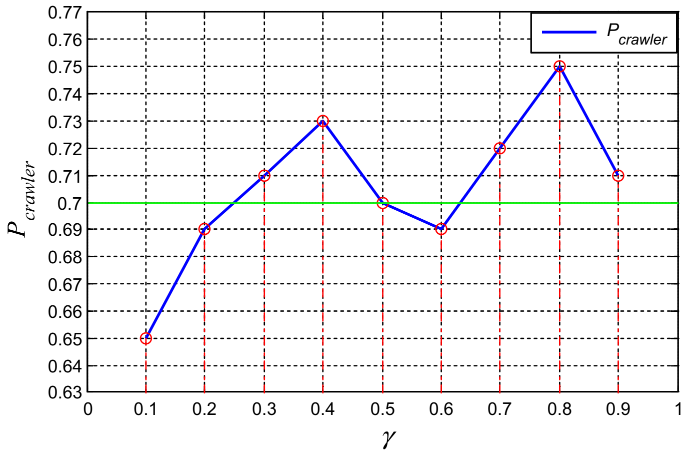


Fig. 2 Precision with different value of parameter γ

4.5.2 The impact analysis of parameter η

Parameter η is one of the important parameters in PC algorithm. It is mainly used to balance the effect of inherited similarity and external similarity correlations between new link and old link. In the experiment for the impact analysis of parameter η, the control of various methods is also adopted to achieve the most suitable value for our proposed algorithm. The value of parameter γ is set to 0.7. The value of parameter η ranges from 0 to 1. Parameter η is determined by $P_{crawler}$ and $R_{crawler}$. $P_{crawler}$ and $R_{crawler}$ are computed based on different values of parameter η, respectively.

Figure 3 describes the value of $P_{crawler}$ by varying the value of parameter η. The value of parameter γ is set to 0.7, and the number of extracted home pages of users is 4000. In this case, with the increase in the value of parameter η, the value of $P_{crawler}$ ranges between 0.71 and 0.83. When $\eta = 0.5$, $P_{crawler}$ achieves highest value 0.83.

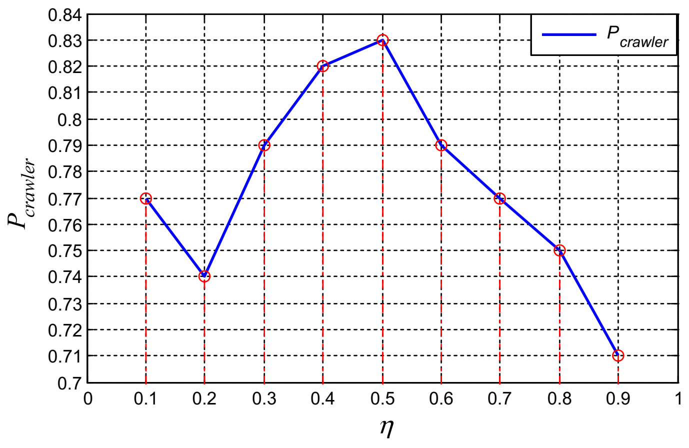


Fig. 3 Precision with different value of parameter η

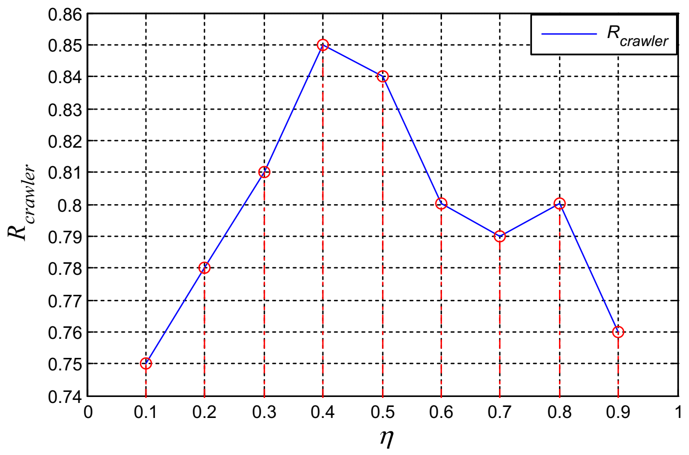


Fig. 4 Recall with different value of parameter η

Therefore, the value of parameter η should be set to 0.5 according to $P_{crawler}$. In this case, the value of parameter η is suitable for PC algorithm.

Figure 4 shows the value of $R_{crawler}$ by varying the value of parameter η. The value of parameter γ is set to 0.7, and the number of extracted home pages of users is 4000. In this case, with the increase in the value of parameter η, the value of $R_{crawler}$ ranges between 0.75 and 0.85. When $\eta = 0.4$, $R_{crawler}$ achieves highest value 0.85. Therefore, the value of parameter η should be set to 0.4 according to $R_{crawler}$. In this case, the value of parameter η is suitable for PC algorithm.

Overall, the value of parameter η should be set to 0.5 according to $P_{crawler}$. But, the value of parameter η should be set to 0.4 according to $R_{crawler}$. Thus, the value of parameter η will be set to 0.45 to better suit PC algorithm by considering both $R_{crawler}$ and $P_{crawler}$.

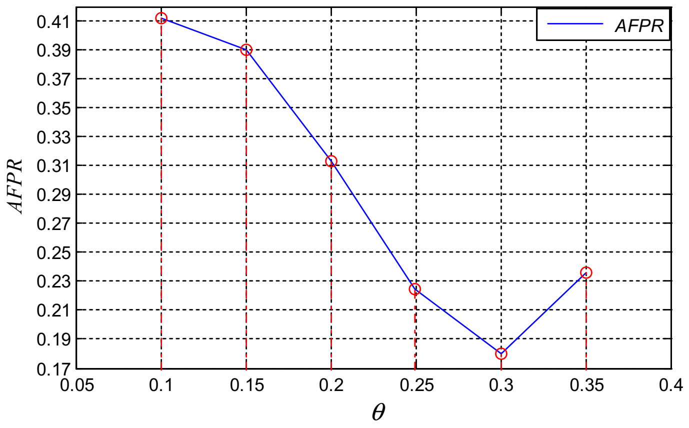


Fig. 5 The average false positive rate with different value of parameter θ

4.5.3 The impact analysis of parameter θ

Parameter θ is one of the important parameters in TC algorithm. It is mainly used to determine the correlation between two items, such as two posts, two topics, one post and one topic, etc. In the experiment for the impact analysis of parameter θ, $AFPR$, $AFNR$ and $\overline{ACEE}$ are used to evaluate the value of parameter θ. Meanwhile, CEE_{AFPR}, CEE_{AFNR} and Pr_{topic} are set to 0.8, 0.2 and 0.01, respectively. ε is the weight of textual information. It is mainly used to balance the effect of textual information and time information correlations between two posts. In the experiment for the impact analysis of parameter θ, ε is set to 0.5.

Figure 5 depicts the value of $AFPR$ by varying the value of parameter θ. ε is set to 0.5. In this case, with the increase in the value of parameter θ, the value of $AFPR$ gets larger earlier and smaller later. When $\theta = 0.3$, $AFPR$ gets smallest value. θ is an important parameter for topic clustering. The larger value of parameter θ will increase the probability that a post is assigned into a topic. Similarly, the smaller value of parameter θ will reduce the probability that a post is assigned into a topic. Thus, the value of parameter θ has an important effect on $AFPR$. Therefore, the value of parameter θ should be set to 0.3 according to $AFPR$, which will be better suitable for TC algorithm.

Figure 6 illustrates the value of $AFNR$ by varying the value of parameter θ. ε is set to 0.5. In this case, with the increase in the value of parameter θ, the value of $AFNR$ gets larger earlier and smaller later. When $\theta = 0.3$, $AFNR$ gets smallest value. θ is an important parameter for topic clustering. The larger value of parameter θ will increase the probability that a post is assigned into a topic. Similarly, the smaller value of parameter θ will reduce the probability that a post is assigned into a topic. Thus, the value of parameter θ has an important effect on $AFNR$. Therefore, the value of parameter θ should be set to 0.3 according to $AFNR$, which will be better suitable for TC algorithm.

Figure 7 describes the value of $\overline{ACEE}$ by varying the value of parameter θ. ε is set to 0.5. In this case, with the increase in the value of parameter θ, the value of

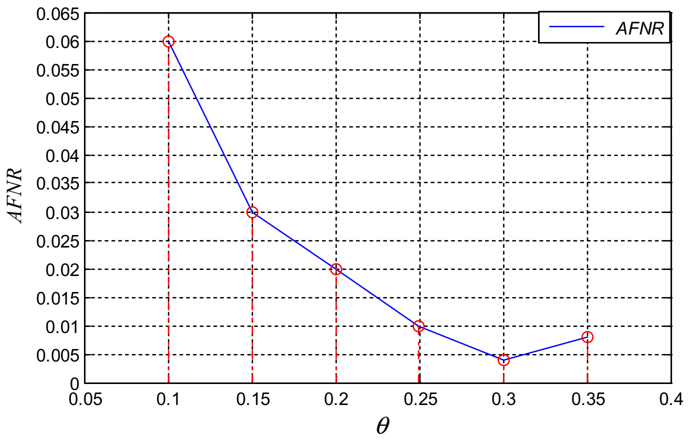


Fig. 6 The average false negative rate with different value of parameter θ

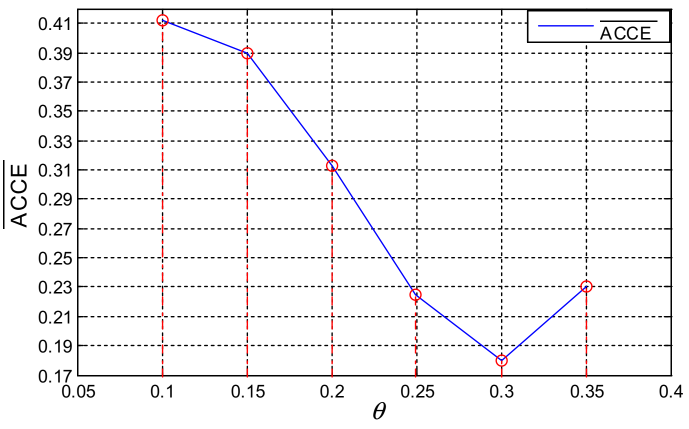


Fig. 7 The normalization of average cost of error execution with different value of parameter θ

$\overline{ACEE}$ gets larger earlier and smaller later. When $\theta = 0.3$, $\overline{ACEE}$ gets smallest value. Therefore, the value of parameter θ should be set to 0.3 according to $\overline{ACEE}$, which will be better suitable for TC algorithm.

Overall, the value of parameter θ should be set to 0.3 according to $AFPR$, $AFNR$ and $\overline{ACEE}$, respectively. Thus, the value of parameter θ will be set to 0.3 to better suit TC algorithm by comprehensively considering $AFPR$, $AFNR$ and $\overline{ACEE}$.

4.5.4 The impact analysis of parameter ε

ε is the weight of textual information. It is mainly used to balance the effect of textual information and time information correlations between two posts. In the experiment for the impact analysis of parameter ε, $AFPR$ and $\overline{ACEE}$ are used to evaluate the

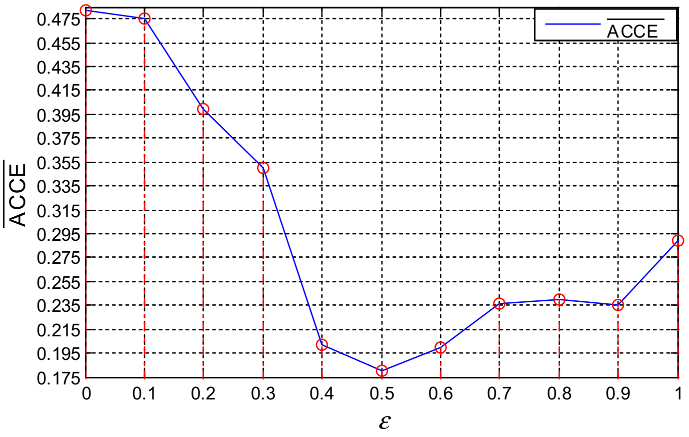


Fig. 8 The normalization of average cost of error execution with different value of parameter ε

value of parameter ε. Meanwhile, CEE_{AFPR}, CEE_{AFNR} and Pr_{topic} are set to 0.8, 0.2 and 0.01, respectively. θ is set to 0.3.

Figure 8 shows the value of $\overline{ACEE}$ by varying the value of parameter ε. With the increase in the value of parameter ε, the value of $\overline{ACEE}$ ranges between 0.18 and 0.482. When $\varepsilon = 0$, $\overline{ACEE}$ achieves the largest value. $\varepsilon = 0$ implies that the similarity between two items only considering the time information. However, the performance of TC algorithm is worst in this case, which means that time information is only an indicator in TC algorithm and it is not enough for TC algorithm to only consider time information. When $\varepsilon = 0.5$, $\overline{ACEE}$ achieves the smallest value. In this case, the performance of TC algorithm is better. Therefore, the value of parameter ε should be set to 0.5 according to $\overline{ACEE}$, which will be better suitable for TC algorithm.

Figure 9 depicts the value of $AFPR$ by varying the value of parameter ε. With the increase in the value of parameter ε, the value of $AFPR$ ranges between 0.18 and 0.543. When $\varepsilon = 0$, $AFPR$ achieves the largest value. $\varepsilon = 0$ implies that the similarity between two items only considering the time information. However, the performance of TC algorithm is worst in this case, which means that time information is only an indicator in TC algorithm and it is not enough for TC algorithm to only consider time information. When $\varepsilon = 0.5$, $AFPR$ achieves the smallest value. In this case, the performance of TC algorithm is better. Therefore, the value of parameter ε should be set to 0.5 according to $AFPR$, which will be better suitable for TC algorithm.

Overall, the value of parameter ε should be set to 0.5 according to $AFPR$ and $\overline{ACEE}$, respectively. Thus, the value of parameter ε will be set to 0.5 to better suit for TC algorithm by comprehensively considering $AFPR$ and $\overline{ACEE}$.

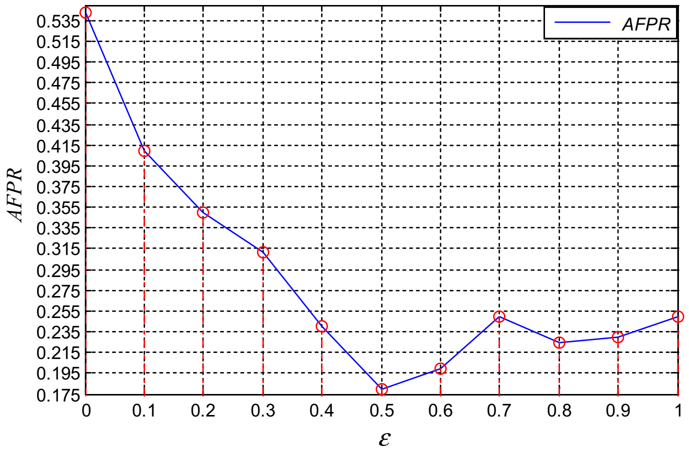


Fig. 9 The average false positive rate with different value of parameter ε

4.6 Comparison and analysis

4.6.1 The comparison and analysis for PC algorithm based on sematic analysis and spatial clustering

Figure 10 illustrates the value of $R_{crawler}$ by varying the number of downloaded links. With the increase in the number of downloaded links, three algorithms have different trends in terms of $R_{crawler}$. For example, the number of downloaded links ranges from 3000 to 4000, the value of $R_{crawler}$ for PC algorithm increases. Meanwhile, the value of $R_{crawler}$ for SS algorithm gets smaller earlier and larger later. But, the value of $R_{crawler}$ for BFS algorithm becomes larger earlier and smaller later. Whatever value the number of downloaded links take, the value of $R_{crawler}$ for PC algorithm is always larger than that of other two algorithms. For instance, when the number of downloaded links is 500, $R_{crawler}$ for PC algorithm is improved up to 5.63% and 8.70% comparing with that for SS algorithm and BFS algorithm, respectively. When the number of downloaded links is 2000, $R_{crawler}$ for PC algorithm can achieve 4.05% and 11.59% improvement over that for SS algorithm and BFS algorithm, respectively. In SS algorithm and BFS algorithm, only the relationship between link and target domain is considered, and the correlation between links is neglected. However, PC algorithm considers not only the relationship between link and target domain, but both the correlation between links and the correlation between old link and new link. Thus, the performance of PC algorithm is better than that of SS algorithm and BFS algorithm in terms of $R_{crawler}$.

Figure 11 shows the value of $P_{crawler}$ by varying the number of downloaded links. With the increase in the number of downloaded links, three algorithms have different trends in terms of $P_{crawler}$. For example, the number of downloaded links ranges from 1500 to 2500, the value of $P_{crawler}$ for PC algorithm gets smaller earlier and larger later. Meanwhile, the value of $P_{crawler}$ for SS algorithm becomes larger earlier

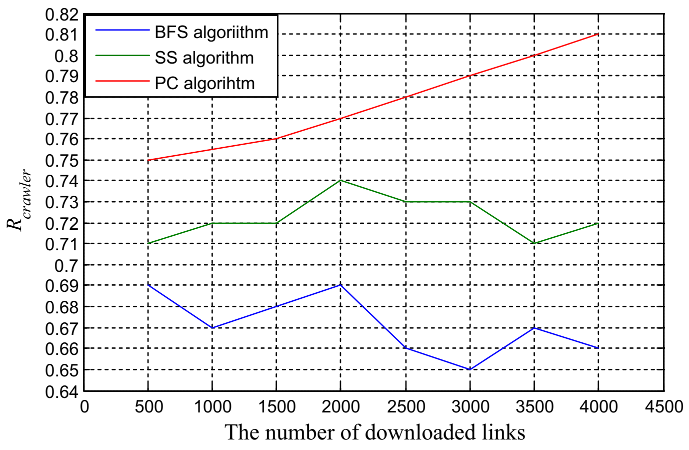


Fig. 10 Recall with different number of downloaded links

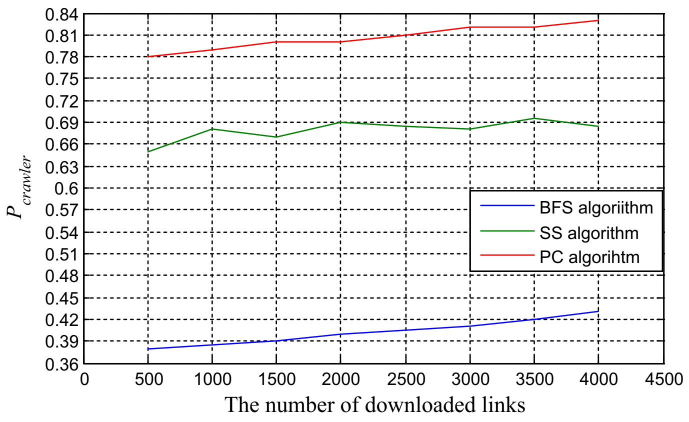
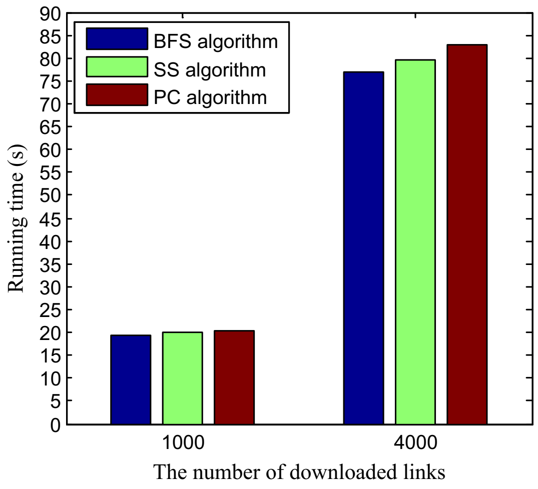


Fig. 11 Precision with different number of downloaded links

and smaller later. But, the value of $P_{crawler}$ for BFS algorithm increases. Whatever value the number of downloaded links take, the value of $P_{crawler}$ for PC algorithm is always larger than that of other two algorithms. For instance, when the number of downloaded links is 500, $P_{crawler}$ for PC algorithm is improved up to 20% and 105.26% comparing with that for SS algorithm and BFS algorithm, respectively. When the number of downloaded links is 4000, $P_{crawler}$ for PC algorithm can achieve 21.17% and 93.02% improvement over that for SS algorithm and BFS algorithm, respectively. In SS algorithm and BFS algorithm, only the relationship between link and target domain is considered, and the correlation between links is neglected. However, PC algorithm considers not only the relationship between link and target domain, but both the correlation between links and the correlation between old link and new link. Thus, the performance of PC algorithm is better than that of SS algorithm and BFS algorithm in terms of $P_{crawler}$.

Fig. 12 Running time with different number of downloaded links



In the experiment for running time with different number of downloaded links, the number of threads is 4. Meanwhile, the download links will be clustered every 100 links. Figure 12 describes the running time by varying the number of downloaded links. With the increase in the number of downloaded links, the running time of each algorithm increases. For example, the running time of PC algorithm with 1000 downloaded links can achieve 306.37% improvement over that of PC algorithm with 4000 downloaded links. Meanwhile, the running time of SS algorithm with 1000 downloaded links can get 297.50% improvement over that of SS algorithm with 4000 downloaded links, and the running time of BFS algorithm with 1000 downloaded links can obtain 300.52% improvement over that of BFS algorithm with 4000 downloaded links. Whatever value the number of downloaded links take, the running time for PC algorithm is always larger than that of other two algorithms. For instance, when the number of downloaded links is 1000, running time for PC algorithm is improved up to 20% and 6.25% comparing with that for SS algorithm and BFS algorithm, respectively. When the number of downloaded links is 4000, the running time for PC algorithm can achieve 4.28% and 7.80% improvement over that for SS algorithm and BFS algorithm, respectively.

In the experiment for download speed with different number of downloaded links, the number of threads is 4. Meanwhile, the download links will be clustered every 100 links. Figure 13 depicts the download speed by varying the number of downloaded links. With the increase in the number of downloaded links, three algorithms have different trends in terms of the download speed. For example, the download speed of PC algorithm with 1000 downloaded links can achieve 1.83% reduction over that of PC algorithm with 4000 downloaded links. Similar, the download speed of BFS algorithm with 1000 downloaded links can achieve 0.19% reduction over that of BFS algorithm with 4000 downloaded links. But, the download speed of SS algorithm with 1000 downloaded links can get 0.6% improvement over that of SS algorithm with 4000 downloaded links. Whatever value the number of downloaded links take, the download speed for PC algorithm is always smaller than that of other two algorithms.

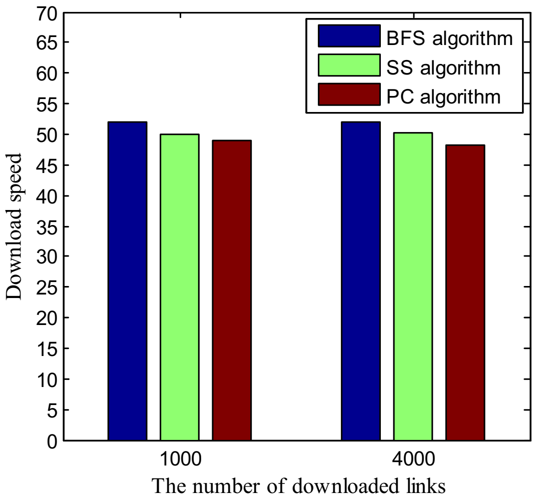


Fig. 13 The download speed with different number of downloaded links

For instance, when the number of downloaded links is 1000, the download speed for PC algorithm is reduced up to 1.80% and 5.76% comparing with that for SS algorithm and BFS algorithm, respectively. When the number of downloaded links is 4000, the download speed for PC algorithm can achieve 4.17% and 7.31% reduction over that for SS algorithm and BFS algorithm, respectively.

Overall, the download speed of BFS algorithm is larger than that of SS algorithm and PC algorithm. However, $R_{crawler}$ and $P_{crawler}$ of BFS algorithm are less than that of SS algorithm and PC algorithm. The download speed of PC algorithm is closer to that of SS algorithm comparing with BFS algorithm. With the increase in the number of downloaded links, the computing process of PC algorithm is more complex due to more considerations. So, the running time of PC algorithm is larger than that of SS algorithm and BFS algorithm. But, $R_{crawler}$ and $P_{crawler}$ of PC algorithm are better than that of SS algorithm and BFS algorithm. In practice, the difference of running time between three algorithms is acceptable. Therefore, the integrity performance of PC algorithm is better than that of SS algorithm and BFS algorithm by synthetically considering $R_{crawler}$, $P_{crawler}$, running time and download speed.

4.6.2 The comparison and analysis for CE algorithm based on document object model tree

Figure 14 shows the comparison results between five participating teams in SEWM 2008 and CE algorithm in dataset I. The Precision and F-measure of CE algorithm and team 1 is better than that of other four algorithms. Moreover, the Precision and F-measure of CE algorithm is slightly better than that of team 1. For example, the Precision of CE algorithm is improved up to 3.60% comparing with that of team 1. F-measure of CE algorithm can achieve 0.30% improvement over that of team 1. The Recall of CE algorithm and team 1 are better than that of other four algorithms. Furthermore, the Recall of CE algorithm is slightly less than that of team 1. For

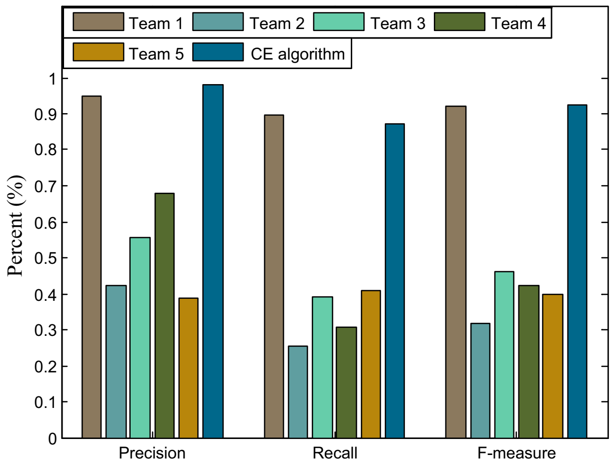


Fig. 14 The comparison results between six algorithms for dataset I

instance, the Recall of CE algorithm is reduced up to 2.60% comparing with that of team 1. Comparing with the content extraction method based on content block segmentation, the content extraction based on labels more suits the actual requirements. CE algorithm can improve the precision of content extraction. Thus, the integrity performance of CE algorithm is better than the compared algorithms.

Figure 15 illustrates the comparison results between five participating teams in SEWM 2008 and CE algorithm in dataset II. The Precision, Recall and F-measure of CE algorithm and team 1 is better than that of other four algorithms. Moreover, the Precision, Recall and F-measure of CE algorithm is better than that of team 1. For example, the Precision of CE algorithm is improved up to 7.90% comparing with that of team 1. The Recall of CE algorithm can achieve 18.57% improvement over that of team 1. F-measure of CE algorithm can get 13.42% improvement over that of team 1. Comparing with the content extraction method based on content block segmentation, the content extraction based on labels more suits for actual need. CE algorithm can improve the precision of content extraction. Thus, the performance of CE algorithm is better than the compared algorithms.

Figure 16 describes the execution time by varying the size of downloaded links. From 0.01 to 5 GB, with the increase in the size of downloaded links, the execution time of Hadoop platform is more than that of multi-process. For example, when the size of downloaded links is 0.1 GB, the execution time of Hadoop platform achieves 88.89% improvement over that of multi-process. Meanwhile, the size of downloaded links is 5 GB, the execution time of Hadoop platform is improved up to 4.09% comparing with that of multi-process. But, with the increase in the size of downloaded links, the difference between Hadoop platform and multi-process reduces. For instance, when the size of downloaded links changes from 0.1 to 5 GB, the improvement for the execution time of Hadoop platform reduces from 88.89 to 4.09%. When the size of downloaded links is 10 GB, the execution time of Hadoop platform is sharply reduced up to 6.43% comparing with that of multi-process. This case implies that

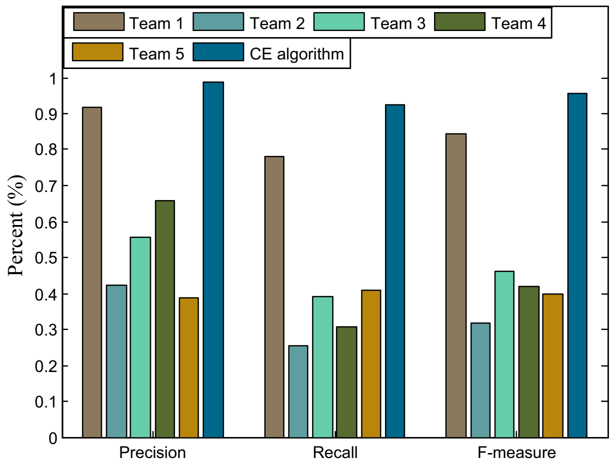


Fig. 15 The comparison results between six algorithms for dataset II

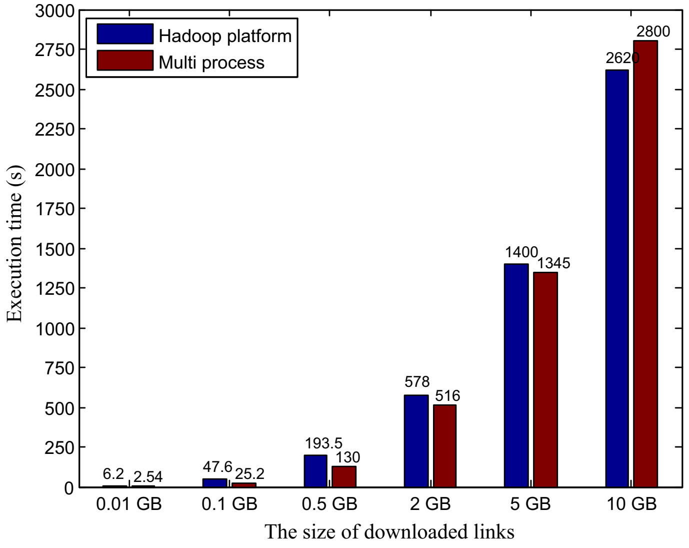


Fig. 16 The execution time with different size of downloaded links

Hadoop platform is suitable for big data process. When the size of data is small, the performance of Hadoop platform is not reflected because most of time of Hadoop platform is taken by the job startup and communication in this case. But, with the increase in size of data, the ratio between the time that is taken by the job startup and communication to overall time reduces. Then, the average time of data process reduces so that the performance of Hadoop platform will be reflected.

Figure 17 shows the execution time per GB by varying the size of downloaded links. From 0.01 to 5 GB, with the increase in the size of downloaded links, the execution time per GB of Hadoop platform is more than that of multi-process. For example,

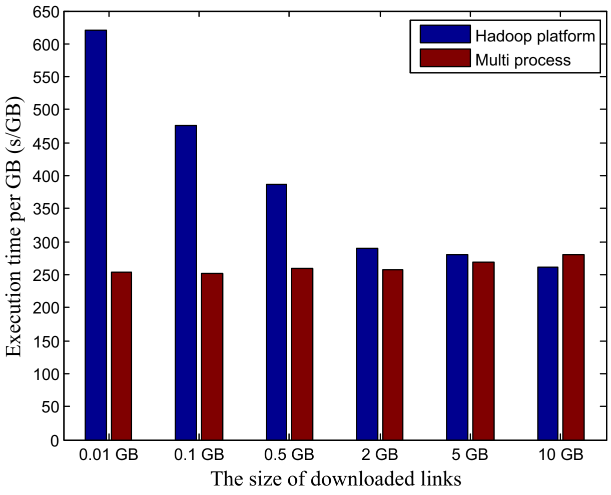


Fig. 17 The execution time per GB with different size of downloaded links

when the size of downloaded links is 0.1 GB, the execution time per GB of Hadoop platform achieves 144.09% improvement over that of multi-process. Meanwhile, the size of downloaded links is 5 GB, the execution time per GB of Hadoop platform is improved up to 4.09% comparing with that of multi-process. But, with the increase in the size of downloaded links, the difference between Hadoop platform and multi-process reduces. For instance, when the size of downloaded links changes from 0.1 to 5 GB, the improvement for the execution time per GB of Hadoop platform reduces from 144.09 to 4.09%. When the size of downloaded links is 10 GB, the execution time per GB of Hadoop platform is sharply reduced up to 6.43% comparing with that of multi-process. This case implies that Hadoop platform is suitable for big data process. When the size of data is small, the performance of Hadoop platform is not reflected because most of time of Hadoop platform is taken by the job startup and communication in this case. But, with the increase in size of data, the ratio between the time that is taken by the job startup and communication to overall time reduces. Then, the average time of data process per GB reduces so that the performance of Hadoop platform will be reflected.

4.6.3 The comparison and analysis for TC algorithm of clusters of new posts and the existed clusters

Figure 18 depicts the value of $AFPR$ by varying the number of posts. With the increase in the number of posts, the value of $AFPR$ ranges between 0.18 and 0.74. When the number of posts increases from 500 to 4000, $AFPR$ sharply reduces. When the number of posts is 4000, $AFPR$ achieves the smallest value. When the number of posts is more than 4000, $AFPR$ is beginning to stabilize. This case implies that when the number of posts is more than 4000, the performance of $AFPR$ in TC algorithm is optimal. Meanwhile, it also indicates that the performance of TC algorithm is better

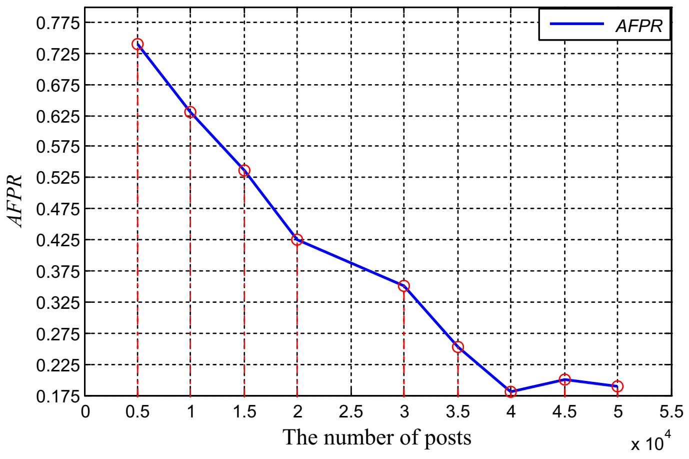


Fig. 18 The average false positive rate with different number of posts

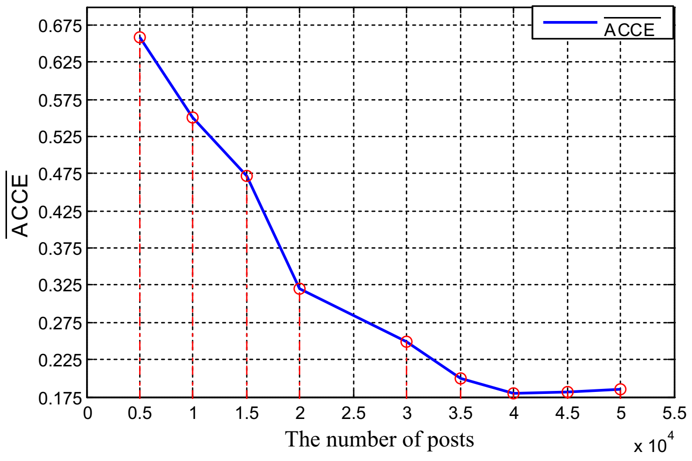


Fig. 19 The normalization of average cost of error execution with different number of posts

with larger number of posts. Therefore, TC algorithm is more suitable for the large-scale social network according to $AFPR$.

Figure 19 illustrates the value of $\overline{ACEE}$ by varying the number of posts. With the increase in the number of posts, the value of $\overline{ACEE}$ ranges between 0.18 and 0.658. When the number of posts increases from 500 to 4000, $\overline{ACEE}$ sharply reduces. When the number of posts is 4000, $\overline{ACEE}$ achieves the smallest value. When the number of posts is more than 4000, $\overline{ACEE}$ is beginning to stabilize. This case implies that when the number of posts is more than 4000, the performance of $\overline{ACEE}$ in TC algorithm is optimal. Meanwhile, it also indicates that the performance of TC algorithm is better with larger number of posts. Therefore, TC algorithm is more suitable for the large-scale social network according to $\overline{ACEE}$.

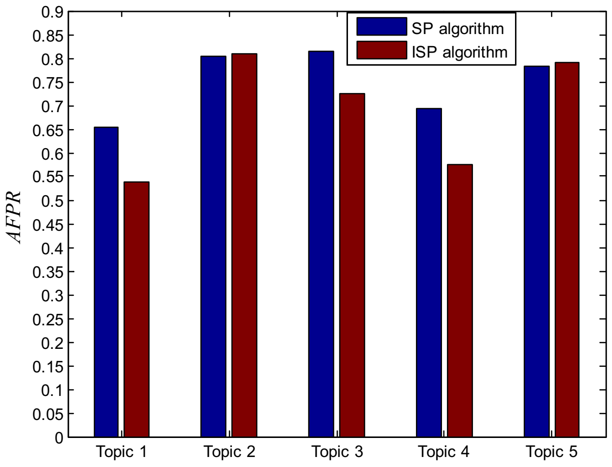


Fig. 20 The average false positive rate with different topics

Overall, according to $AFPR$ and $\overline{ACEE}$, TC algorithm is more suitable for the large-scale social network. Thus, the integrity performance of TC algorithm is better in processing large-scale social network.

For evaluating the performance of TC algorithm, 117 thousand and 130 posts in forum are extracted and analyzed by CE algorithm. Then, these posts are clustered into 8976 topics. The top five topics with most posts are "Vladimir Putin missing," "Malaysian jetliner missing," "Jackie Chan," "Spring Festival Gala" and "Li Guangyao death," which can be named topic 1 to 5, respectively. These topics include 98,650, 93,481, 83,256, 89,735 and 5425 posts, respectively.

Figure 20 describes the value of $AFPR$ with different topics. Figure 20 indicates that the performance of ISP algorithm is better than that of SP algorithm. For example, in "Vladimir Putin missing" topic, $AFPR$ of ISP algorithm achieves 17.72% reduction over that of SP algorithm. Also, in "Spring Festival Gala" topic, the value of $AFPR$ with ISP algorithm is reduced up to 17.19% comparing with SP algorithm. But, in "Li Guangyao death" topic, the value of $AFPR$ with ISP algorithm is improved up to 0.71% comparing with SP algorithm. This is because "Li Guangyao death" topic only includes 5425 posts and the ISP algorithm is more suitable to process large-scale social network. Though the performance of ISP algorithm is less than that of SP algorithm in small-scale social network, the difference between them is small. Therefore, in aggregate the performance of ISP algorithm is better than that of SP algorithm.

Figure 21 illustrates the comparison results between three TC algorithms. The performance of K-mean algorithm, non-parallel ISP algorithm and parallel ISP algorithm are compared in terms of $AFPR$, $AFNR$ and $\overline{ACCE}$. The results show that the value of $AFPR$, $AFNR$ and $\overline{ACCE}$ in K-mean algorithm are always largest than that in both non-parallel ISP algorithm and Parallel ISP algorithm. For example, $AFPR$ of K-mean algorithm can achieve 67.14% and 83.77% improvement over that of non-parallel ISP algorithm and parallel ISP algorithm, respectively. Meanwhile, $AFNR$ of K-mean algorithm is improved up to 157.14% and 350% comparing with that of

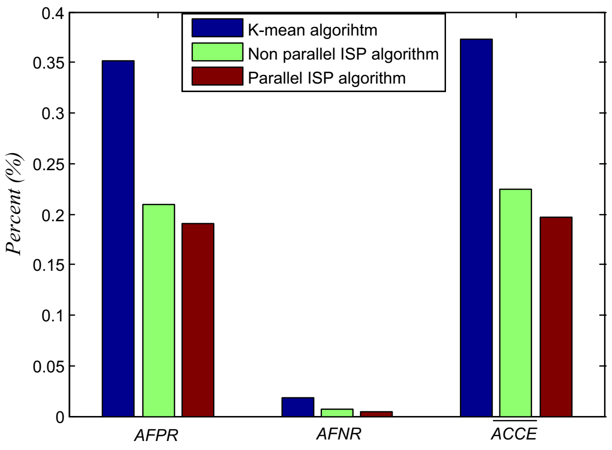


Fig. 21 The comparison results between three TC algorithms

non-parallel ISP algorithm and parallel ISP algorithm, respectively. Thus, the performance of K-mean algorithm for topic clustering is worst than that of both non-parallel ISP algorithm and parallel ISP algorithm. Furthermore, the values of $AFPR$, $AFNR$ and $\overline{ACCE}$ in non-parallel ISP algorithm are always larger than that in Parallel ISP algorithm. For instance, $AFPR$ of non-parallel ISP algorithm can achieve 9.95% improvement over that of parallel ISP algorithm. Meanwhile, $AFNR$ of non-parallel ISP algorithm is improved up to 75% comparing with that of parallel ISP algorithm. Therefore, the performance of parallel ISP algorithm for topic clustering is better than that of non-parallel ISP algorithm. Likewise, the performance of parallel ISP algorithm for topic clustering is best than that of K-mean algorithm and non-parallel ISP algorithm. So, it is reasonable that ISP algorithm runs on Hadoop platform.

5 Conclusion

In this section, a crawler model based on semantic analysis and spatial clustering is proposed firstly. Then, the content extraction model based on document object model tree is built to extract the target text information from the links fetched by the proposed crawler model. The similarities between textual information in different regions are computed to choose the important information. Moreover, a two-stage topic clustering model based on time information is presented. The time information is introduced into the similarity computation between two posts or clusters. The single-pass algorithm is improved to be applied in different clustering stage to improve the clustering accuracy. Finally, the proposed algorithms are evaluated on Hadoop platform. The Hadoop platform can effectively reduce the computing time and improve the server quality of users in large-scale social network. Meanwhile, the experiments demonstrate that the proposed algorithms are suitable for the data process in large-scale social network.

In this paper, the CE algorithm is mainly used to extract the text information in large-scale social network. For illustrating the application scenario of CE algorithm, the forum is regarded as an example. Furthermore, ISP algorithm is proposed to improve the performance of topic clustering. Thus, ISP algorithm is more suitable for topic clustering in forum, which is also the limitation or shortcoming of our paper.

So, in subsequent work, we will test and verify the proposed algorithms in more data sets and application scenario. Utilizing the proposed algorithms to track the evolution of public opinion would be a promising future research direction.

Acknowledgements The work was supported by the National Natural Science Foundation (NSF) under grants (No. 61672397, No. 61873341, No. 61472294, No. 61771354), Application Foundation Frontier Project of WuHan (No. 2018010401011290). Open Fund of Key Laboratory of Urban Land Resources Monitoring and Simulation, Ministry of Land and Resources (Grant No. KF-2018-03-005). Key Lab of Guangdong for Utilization of Remote Sensing and Geographical Information System, Guangzhou Institute of Geography (Grant No. 2017B030314138). Jiangsu Key Laboratory of Meteorological Observation and Information Processing, Nanjing University of Information Science and Technology (Grant No. KDXS1804). Any opinions, findings, and conclusions are those of the authors and do not necessarily reflect the views of the above agencies.

References

1. Akhgar B, Saathoff GB, Arabnia HR, Hill R, Staniforth A, Bayerl PS (2015) Application of big data for national security: a practitioner's guide to emerging technologies. Butterworth-Heinemann, Oxford
2. Arabnia HR, Fang WC, Lee C et al (2010) Context-aware middleware and intelligent agents for smart environments. IEEE Intell Syst 25(2):10–11
3. Salton G, Wong A, Yang CS (1974) A vector space model for automatic indexing. Commun ACM 18(11):613–620
4. Hull DA (1996) Stemming algorithms: a case study for detailed evaluation. J Assoc Inf Sci Technol 47(1):1–27
5. Fortunato S, Barthélemy M (2007) Resolution limit in community detection. Proc Natl Acad Sci USA 104(1):36–41
6. Hassan T, Cruz C (2017) Ontology-based approach for unsupervised and adaptive focused crawling. In: International Workshop on Semantic Big Data. ACM (2), pp 1–24
7. Bai S, Hussain S, Khoja S (2016) A framework for focused linked data crawler using context graphs. In: International Conference on Information and Communication Technologies. IEEE, pp 1–6
8. Gupta S (2016) Design of focused crawler based on feature extraction, classification and term extraction. In: International Conference on Computing for Sustainable Global Development. IEEE, pp 3429–3434
9. Vieira K, Barbosa L, Silva AS et al (2016) Finding seeds to bootstrap focused crawlers. World Wide Web-internet Web Inf Syst 19(3):449–474
10. Almuhareb A (2016) Arabic poetry focused crawling using SVM and keywords. In: Saudi International Conference on Information Technology. IEEE, pp 1–4
11. Du Y, Liu W, Lv X et al (2015) An improved focused crawler based on Semantic Similarity Vector Space Model. Appl Soft Comput 36(C):392–407
12. Wei Y, Li P (2018) Designing focused crawler based on improved genetic algorithm. In: Tenth International Conference on Advanced Computational Intelligence. IEEE, pp 319–343
13. Boukadi K, Rekik M, Rekik M et al (2018) FC4CD: a new SOA-based focused crawler for cloud service discovery. Computing 6:1–27
14. Pouriyeh S, Allahyari M, Kochut K et al (2018) Combining word embedding and knowledge-based topic modeling for entity summarization. In: IEEE, International Conference on Semantic Computing. IEEE Computer Society, pp 252–255
15. Luper D, Cameron D, Miller JA, Arabnia HR (2007) Spatial and temporal target association through semantic analysis and GPS data mining. In: Proceedings of 2007 International Conference on Information and Knowledge Engineering (IKE'07), USA, pp 251–257
16. Zhang J, Ding WZ (2016) An improved ontology-based web information extraction. In: Educational Innovation Through Technology. IEEE, pp 37–41

17. Fagin R, Kimelfeld B, Reiss F et al (2014) Cleaning inconsistencies in information extraction via prioritized repairs. ACM 23:164–175
18. Velasco-Elizondo P, Marín-Piña R, Vazquez-Reyes S et al (2016) Knowledge representation and information extraction for analyzing architectural patterns. Sci Comput Program 121:176–189
19. Gao B, Zhu J et al (2016) High-quality information extraction and query-oriented summarization for automatic query-reply in social network. Expert Syst Appl Int J 44(C):92–101
20. Mehdi A, Seyedamin P, Krys K, Hamid RA (2017) A knowledge-based topic modeling approach for automatic topic labeling. Int J Adv Comput Sci Appl 8(9):335–349
21. Seyedamin P, Mehdi A, Krys K, Gong C, and Hamid RA (2017) ES-LDA: entity summarization using knowledge-based topic modeling. In: Proceedings of the Eighth International Joint Conference on Natural Language, pp 316–325
22. Yeh JF, Tan YS, Lee CH (2016) Topic detection and tracking for conversational content by using conceptual dynamic latent Dirichlet allocation. Neurocomputing 216:310–318
23. Lin H, Sun B, Wu J et al (2016) Topic detection from short text: a term-based consensus clustering method. In: International Conference on Service Systems and Service Management. IEEE, pp 1–6
24. Chakraborti S, Dey S (2016) Multi-level K-means text clustering technique for topic identification for competitor intelligence. In: IEEE Tenth International Conference on Research Challenges in Information Science. IEEE, pp 1–11
25. Hashimoto K, Kontonatsios G, Miwa M et al (2016) Topic detection using paragraph vectors to support active learning in systematic reviews. J Biomed Inform 62(C):59–65
26. Zhang C, Wang H, Cao L et al (2016) A hybrid term–term relations analysis approach for topic detection. Knowledge-Based Syst 93:109–120
27. Nguyen KL (2016) Hot topic detection and technology trend tracking for patents utilizing term frequency and proportional document frequency and semantic information. In: International Conference on Big Data and Smart Computing. IEEE, pp 223–230
28. Mehta B, Narvekar M (2015) DOM tree-based approach for Web content extraction. In: International Conference on Communication, Information and Computing Technology. IEEE, pp 1–6
29. Bisson M, Bernaschi M, Mastrostefano E (2016) Parallel distributed breadth first search on the Kepler architecture. IEEE Trans Parallel Distrib Syst 27(7):2091–2102
30. Qiu L, Lou Y, Chang M (2016) Research on theme crawler based on Shark-Search and PageRank algorithm. In: International Conference on Cloud Computing and Intelligence Systems. IEEE, pp 268–271
31. Shahrivari S, Jalili S (2016) Single-pass and linear-time k-means clustering based on MapReduce. Elsevier, Amsterdam