A NOVEL APPROACH BASED ON TOPIC MODELING FOR CLONE GROUP MAPPING

Ruixia Zhang, Liping Zhang, Huan Wang and Zhuo Chen

Computer and information engineering college,
Inner Mongolia normal university, Hohhot, China
zhangruixia923@163.com

ABSTRACT

Clone group mapping has a very important significance in the evolution of code clone. The topic modeling techniques were applied into code clone firstly and a new clone group mapping method was proposed. By using topic modeling techniques to transform the mapping problem of high-dimensional code space into a low-dimensional topic space, the goal of clone group mapping was indirectly reached by mapping clone group topics. Experiments on four open source software show that the recall and precision are up to 0.99, thus the method can effectively and accurately reach the goal of clone group mapping.

KEYWORDS

code clone; software evolution; topic; topic modeling; clone group mapping

1. INTRODUCTION

The activities of the programmers including copy, paste and modify result in lots of code clone in the software systems. A code clone is a code portion in source files that is identical or similar to another[1]. It is suspected that many large systems contain approximately 9%-17% clone code, sometimes as high as even 50%[2].

After a decade of active research, it is evident that code clones have both a positive [3] and a negative [4] impact in the maintenance and evolution of software systems. For example, copying source code without defect can reduce the potential risk of writing new code, save development time and cost; code clones can cause additional maintenance effort. Changes to one segment of code may need to be propagated to several others, incurring unnecessary maintenance costs.

Code clone is inevitable in software development, and in order to exploit the advantages of clones while lowering their negative impact, it is important to understand the evolution of clones and manage them accordingly. Therefore， in order to meet the demands of clone evolution, an clone mapping method is put forward. Clone group mapping reflect how an clone group evolve from a previous version to the current version, is the core technology in the evolution of code clone across versions.

The topic modeling techniques is applied to code clone firstly and a new clone group mapping method is proposed. Topic modeling technology can make full use of the source text and structure information to transform the mapping problem of high-dimensional code space into a

Dhinaharan Nagamalai et al. (Eds) : CCSEA, DKMP, AIFU, SEA - 2015
pp. 213–222, 2015. © CS & IT-CSCP 2015 DOI : 10.5121/csit.2015.50218

low-dimensional topic space, and thus indirectly achieve the clone group mapping purposes by mapping clone group topics.

2. RELATED WORK

Software development and maintenance in practice follow a dynamic process. With the growth of the program source, code clones also experience evolution from version to version. what change the Clone group have happened from one version to next need to be made judgments by clone group mapping.

To map clone group across consecutive versions of a program, mainly five different approaches have been found in the literature.

- **Based on text[5]：** It separates clone detection from each version, and then similarity based heuristic mapping of clones in pairs of subsequent versions. Text similarity are often computed by the Longest Common Subsequence(LCS)or Edit Distance(Levenshtein Distance, LD) algorithm that have quadratic runtime, which lead to inefficient clone mapping. The method is susceptible to large change in clone.

- **Based on version management tools (CVS or SVN)[6]：** Clones detected from the first version are mapped to consecutive versions based on change logs obtained from source code repositories. It is faster than the above technique, but can miss the clones introduced after the first version.

- **Based on incremental clone detection algorithm[7]：** Clones are mapped during the incremental clone detection that used source code changes between revisions. It can reduce the redundant computation and save time .So it is faster than the above two techniques, but cannot operate on the clone detection results obtained from traditional non-incremental tools.

- **Based on functions[8]:** It separates clone detection from each version, functions are mapped across subsequent versions, then clones are mapped based on the mapped functions. To some extent, it improves the efficiency and accuracy of the mapping, but it is susceptible to similar overloaded/overridden functions for its over-reliance on function information.

- **Based on CRD(Clone Region Descriptor)[9]：** Clone code is represented by CRD, then clones are mapped based on CRD between versions. It is not easily influenced by position of the code clone. Mutations or big difference between versions can reduce the mapping validity greatly.

This paper presents a new clone group mapping method based on topic modeling technology, unlike the mapping method based on text, the basic collection of the mapping is the clone group topics, not intermediate representation of clone code(e.g., token and AST). Topic has a large granularity and the higher level of abstraction. However, the difference of topics between different clone groups in the same version is very large and the difference of topics between same clone groups in the different version is very little, which make the clone group mapping method based on topic modeling practicable and effective.

3. APPROACH

In this section we present a new clone group mapping approach based on topic modeling for tracking clone groups across different versions.

3.1 Overview of Topic Model

Topic models are generative probabilistic models, originally used in the area of natural language processing for representing text documents. LDA (Latent Dirichlet Allocation) has recently been applied to a variety of domains, due to its attractive features. First, LDA enables a low-dimensional representation of text, which (i) uncovers latent semantic relationships and (ii) allows faster analysis on text [10]. Second, LDA is unsupervised, meaning no labeled training data is required for it to automatically discover topics in a corpus. And finally, LDA has proven to be fast and scalable to millions of documents or more [11]. For these reasons, in this paper we use LDA as our topic model.

In the LDA model, LDA is statistical models that infer latent topics to describe a corpus of text documents [12]. Topics are collections of words that co-occur frequently in the corpus. For example, a topic discovered from a newspaper collection might contain the words {cash bank money finance loan}, representing the “finance industry” concept; another might contain {fish river stream water bank}, representing the “river” concept. So, documents can be represented by the topics within them. Topic modeling techniques transform the text into topic space.

Recently, researchers found topics to be effective tools for structuring various software artifacts, such as source code, requirements documents, and bug reports. Kuhn [13] made the first attempt to apply topic modeling technique to source code, and tried to discover the functional topics. W. Thomas[14]performed a detailed investigation of the usefulness of topic evolution models for analyzing software evolution, they found that topic models were an effective technique for automatically discovering and summarizing software change activities. Asuncion[15] used the topic modeling techniques to study software traceability. Tian [16]used LDA to Automatically classify software in the software repository. Gethersd[17]developed the IDE plug-in, they combined topic modeling results and the existing software development tools to help developers apply topic modeling results. Liu Chao[18] applied topic model to retrieval traceability links between source code and Chinese documentation. Xie Bing[19] from Beijing university proposed a function recognition approach based on LDA and code static analysis technology o better support the activity of code reuse. In addition, topic model was also used to study class cohesion [20]and bug location[21].

3.2 Mapping Clone Group Based on Topic Modeling

3.2.1 Framework of The Algorithm.

Typically a clone detection tool reports results as a collection of clone groups where each clone group has two or more clone fragments. The paper uses the LDA topic modeling technology to map clone group. It mainly works in the following three steps: (1) extracting the topics from clone group, (2) calculating the similarity between topics,(3) mapping clone group topics. Let $\mathrm{CG}^{\mathrm{n}} = \{\mathrm{cg}_1^{\mathrm{n}}, \mathrm{cg}_2^{\mathrm{n}}, \cdots \mathrm{cg}_{\mathrm{s}}^{\mathrm{n}}\}$be the reported clone groups in V_{n}, $\mathrm{T}^{\mathrm{n}} = \{\mathrm{t}_1^{\mathrm{n}}, \mathrm{t}_2^{\mathrm{n}}, \cdots \mathrm{t}_{\mathrm{s}}^{\mathrm{n}}\}$refers to the clone group topics extracted from the clone group $\mathrm{CG}^{\mathrm{n}}$where $\mathrm{t}_{\mathrm{i}}^{\mathrm{n}}$ was extracted from $\mathrm{cg}_{\mathrm{i}}^{\mathrm{n}}$,$1 \leq \mathrm{i} \leq \mathrm{n}$. Figure 1 shows the framework of the algorithm.

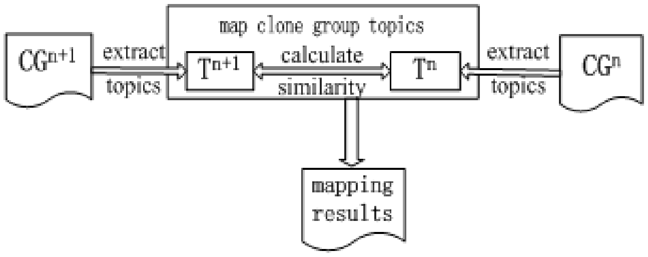


Figure 1. The frame of mapping algorithm

To track clone groups over two different versions V_{n+1} and V_n,we compare every clone group in version $V_{n+1}$to every clone group in version V_n. Topic modeling technology is used to extract the topics from each clone group in the version V_n and V_{n+1}respectively. At this point, since topic is the only representation of corresponding clone group, the problem of mapping clone group between two versions of a program is reduced to the mapping of clone group topics between two versions. Then clone group topics are mapped by comparing similarity between topics in the version V_{n+1} and V_n. If the topic t_i^{n+1} of a clone group cg_i^{n+1} in version v_{n+1} matches to the topic t_j^n of a clone group cg_j^n in the version V_n, we know that the clone group cg_i^{n+1} in V_{n+1} and the clone group cg_j^n in V_n are the same. Due to the transitivity of the relation of equivalence， we can then conclude that clone group cg_i^{n+1} is related to clone group cg_j^n.The algorithm is as follows:

Clone Group Mapping algorithm

```
● $\forall\ cg_i^{n+1} \in CG^{n+1}, CG^{n+1}\ in\ V_{n+1}$
    ● extract $t_i^{n+1}$ from $cg_i^{n+1}$
    ● $\forall\ cg_j^n \in CG^n, CG^n\ in\ V_n$
        ● extract $t_j^n$ from $cg_j^n$
        ● caculate similarity between $t_j^n$ and $t_i^{n+1}$, and store similarity value in the array
          unit sim [j]
    ● suppose sim[k]=max{sim[ ]}:
      IF sim[k]$\geqslant\delta$， $cg_i^{n+1}$ is mapped back to $cg_j^n$ ,namely $cg_i^{n+1}$—> $cg_j^n$ ；
     THEN $cg_i^{n+1}$—— >null.
● Return all mapping results
```

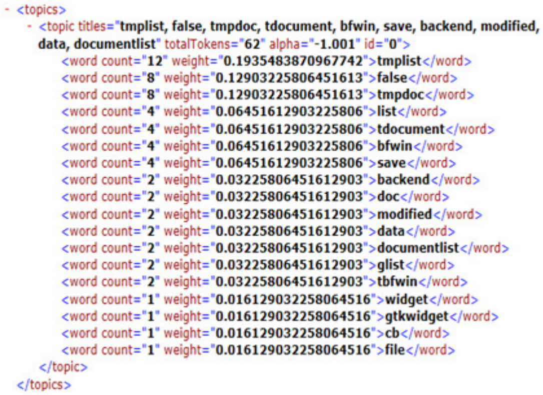
3.2.2 Extract Clone Group Topics

Under the standard programming style, software is suitable for extracting the topics using the LDA model. The paper uses MALLET topic modeling toolkit to extract the topics. MALLET is a Java-based package for statistical natural language processing, document classification, clustering, topic modeling, information extraction, and other machine learning applications to text. It contains efficient, sampling-based implementations of LDA.

Preprocess the source code. Each topic is collections of words that co-occur frequently in the clone group, and is the only representation of corresponding clone group. The topic contains a large number of stop words which play a small role in characterization of clone group information. So, we remove stop words to reduce noise before extracting the topics. Stop words mainly include the following three categories : 1） programming language keywords, such as "for", "return", and "class", etc. 2) Programming related words, such as "main", "arg", and "util", etc. 3） common English language stop words, such as "the", "it", and "on", etc.

Choose the number of topic. In order to make the topic accurately represent the information of clone group, the proper number of topic is a key to influence the accuracy of clone group mapping. For any given corpus, there is no provably optimal choice for the number of topics. The choice is a trade-off between coarser topics and finer-grained topics. setting the number of topics to extremely small values results in topics that contain multiple concepts, while setting the number of topics to extremely large values results in topics that are too fine to be meaningful and only reveal the idiosyncrasies of the clone group.

In the paper, through experimental analysis, it is best for setting the number of topics to one. In the same clone group, clone code is a code portion that is identical or similar to another. The whole clone group is multiple copies of the same clone whose syntactic or semantic function is same. In other words, a clone group can be represented by a topic. The topics extracted from clone group by MALLET are shown in Figure 2.



```
- <topics>
  - <topic titles="tmplist, false, tmpdoc, tdocument, bfwin, save, backend, modified,
    data, documentlist" totalTokens="62" alpha="-1.001" id="0">
      <word count="12" weight="0.1935483870967742">tmplist</word>
      <word count="8" weight="0.12903225806451613">false</word>
      <word count="8" weight="0.12903225806451613">tmpdoc</word>
      <word count="4" weight="0.06451612903225806">list</word>
      <word count="4" weight="0.06451612903225806">tdocument</word>
      <word count="4" weight="0.06451612903225806">bfwin</word>
      <word count="4" weight="0.06451612903225806">save</word>
      <word count="2" weight="0.03225806451612903">backend</word>
      <word count="2" weight="0.03225806451612903">doc</word>
      <word count="2" weight="0.03225806451612903">modified</word>
      <word count="2" weight="0.03225806451612903">data</word>
      <word count="2" weight="0.03225806451612903">documentlist</word>
      <word count="2" weight="0.03225806451612903">glist</word>
      <word count="2" weight="0.03225806451612903">tbfwin</word>
      <word count="1" weight="0.016129032258064516">widget</word>
      <word count="1" weight="0.016129032258064516">gtkwidget</word>
      <word count="1" weight="0.016129032258064516">cb</word>
      <word count="1" weight="0.016129032258064516">file</word>
    </topic>
  </topics>
```

Figure 2.The topics extracted by MALLET

3.2.3 Mapping Clone Group Topics

Clone group mapping is determined by the degree of similarity between clone groups in the different versions. In the paper, clone group mapping is determined indirectly by similarity between clone group topics from different versions. If the similarity between the topic t_j^n and topic $t_i^{n+1}$is highest, and the similarity values is not less than certain threshold (δ).In that way, we can conclude that the topic t_i^{n+1} is mapped back to the topic t_j^n, namely the clone group $cg_i^{n+1}$is mapped back to the clone group cg_j^n . In the paper, Similarity threshold δ is set to 0.8. That's because the similarity value between $t_i^{n+1}$and t_j^n vary from 0.8 to 1 when cg_i^{n+1} is mapped back to the clone group cg_j^n, and the similarity value between $t_i^{n+1}$and t_j^n is less than 0.8 when cg_j^n is not origin of clone group cg_i^{n+1}.

In the paper, the mapping is carried out from the version $V_{n+1}$to V_n. That's because the number of clone group is generally on the rise in the process of software evolution. If the mapping is carried out from the version V_n to V_{n+1}, new clone groups are failed to map. On the contrary, disappeared clone groups are failed to map. However, we are more interested in clone code near to the current version in the study of clone evolution. That is to say, compared with disappeared clone group, we are more interested in new clone group. So, the mapping is carried out from the version $V_{n+1}$to V_n.

4. CASE STUDY

4.1 Systems Under Study

Due to the difference in size of software system, number of clone group in each version ranging from dozens to thousands ,in view of the limitations of manually inspection, so we perform case study on the source code of four small and medium-sized, open source software systems which is written in different programming languages. The detail of software is shown in Table 1.

Table 1. The detail of software

software	Bluefish	MALLET	ArgoUML	PostgreSQL
Implementation language	C	JAVA	JAVA	C
Average size	23MB	31MB	35MB	92MB
Number of the selected version	2	3	3	4
Number of Clone group (on average)	20	145	299	506

In the paper, NiCad is used to detect clone code. NiCad , a clone detector, can detect Type-1 、Type-2 and Type-3 clones written in multiple programming language（C、JAVA、C#）and have a high precision rate and recall rate. In the Linux platform, NiCad is used to detect Type 1, Type 2 and Type 3 clones of the software. Then we transfer clone group files to the Windows platform to map the clone group across versions.

4.2 Evaluation Measures

To evaluate the feasibility and validity of the approach, we use Precision and Recall as Evaluation Measures to manually inspect the results of the approach based on topic modeling . Precision and Recall are defined as follows:

Precision: Of all the clone group mappings discovered, how many are correct?
We calculate the precision of the experimental result as

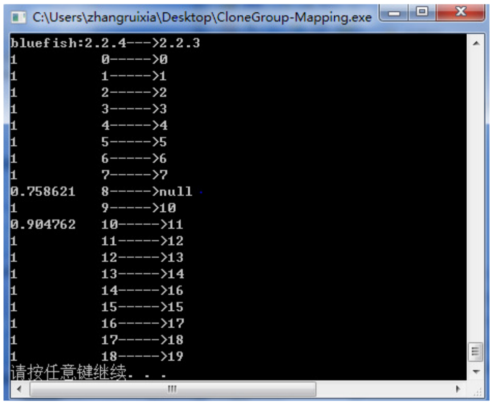
$$\text{Precision} = \frac{\text{the number of correct mapping}}{\text{the number of correct and incorrect mapping}}$$

Recall: Of all the actual clone group mappings, how many were discovered?
We calculate the recall of the experimental result as

$$\text{Recall} = \frac{\text{the number of correct mapping}}{\text{the number of actual clone group mappings}}$$

4.3 Results

Take bluefish for example, mapping results of clone groups between bluefish 2.2.4 and bluefish 2.2.3 are shown in Figure 3.



```
bluefish:2.2.4--->2.2.3
1           0----->0
1           1----->1
1           2----->2
1           3----->3
1           4----->4
1           5----->5
1           6----->6
1           7----->7
0.758621    8----->null
1           9----->10
0.904762    10----->11
1           11----->12
1           12----->13
1           13----->14
1           14----->16
1           15----->15
1           16----->17
1           17----->18
1           18----->19
请按任意键继续. . .
```

Figure 3. Mapping results of clone groups between Bluefish 2.2.4 and 2.2.3

The second and third column of the figure show clone group number of the corresponding version in Bluefish. There are 19(From 0 to 18) clone groups in Bluefish 2.2.4. There are 20(From 0 to 19) clone groups in Bluefish 2.2.3.The arrows indicate the corresponding clone group is traced to its origin clone group. For example, the 14th clone group of Bluefish 2.2.4 is mapped back to the 16th clone group of Bluefish 2.2.3.But the 14th clone group of Bluefish 2.2.4 is not traced to its origin clone group, which indicate that it is a new clone group, probably the great changes have taken place in its origin during the software evolution from Bluefish 2.2.3 to Bluefish 2.2.4, which similarity value between them is less than the threshold δ. We note that 8th and 9th clone group of Bluefish2.2.3 do not appear in the list, probably they are removed or take great change during software evolution. The first column of the figure show the largest similarity values of clone group topics between Bluefish 2.2.4 and Bluefish 2.2.3. If the value is not less than δ (0.8), There is a mapping relationship between them. It can be seen in the Figure 3 that most of the similarity value are as high as 1, namely most of the clone groups do not change during software evolution. Few of the similarity values are not 1, which indicate that clone codes have experienced some degree of change, such as the clone group is deleted, a few of clone fragments are added or removed.

Clone group mapping is carried on consecutive versions of other software, and manually inspect the Precision and Recall of the mapping results. The results can be seen in Table 2 and Table 3. The Precision and Recall of the approach are as high as 0.99, which reveal the validity and feasibility of clone group mapping approach based on topic modeling. The runtime of clone group mapping across versions is acceptable. Since number of clone groups in some software is large, in view of the limitations of manually inspection, we conduct experiments on only 12 versions of the above 4 software. But the results are enough to reveal the feasibility of the approach.

Table 2. The experimental results of the approach

Software and versions	**Bluefish**		**PostgreSQL**		**PostgreSQL**		**PostgreSQL**	
Evaluation Measures	2.2.4	2.2.3	9.1.5	9.1.4	9.1.4	9.1.3	9.1.3	9.1.2
Precision	1		0.996		0.996		0.994	
Recall	0.95		1		1		1	

Table 3. The experimental results of the approach

Software and versions	ArgoUML		ArgoUML		MALLET		MALLET	
Evaluation Measures	0.27.3	0.27.2	0.27.2	0.27.1	2.0.7	2.0.6	2.0.6	2.0.5
Precision	1		0.996		1		0.992	
Recall	0.996		0.993		0.982		0.992	

5. DISCUSSION AND THREATS TO VALIDITY

5.1 Limitations of Similarity Threshold

In the paper, similarity threshold between clone group topics across versions is determined based on the experience knowledge, and different software use the same similarity threshold, which have an impact on the results. Firstly, similarity threshold based on the experience knowledge can't reflect mapping efficiency of the algorithm. Secondly, the same threshold is used to different software that they exist remarkable differences in programming language, programming style and the degree of change between versions, which will reduce the validity of the mapping algorithm.

5.2 Limitations of Clone Detector

The clone detector provides the basis data for clone group mapping, so clone group mapping approach directly is affected by clone detector. It is critical for clone group mapping to choose an accurate clone detector.

5.3 The Differences between Versions

It is discovered by the experimental results that the smaller differences between versions is, the higher accuracy the approach has. If clone group have happened so significant changes during software evolution that similarity value between two versions exceed the permitted threshold, which clone group that could have been traced to its origin clone group is failure to mapping. That is to say, Mutation or big difference between versions can reduce the accuracy of the mapping. Therefore, It contributes to improvement of accuracy of clone group mapping that using revision of software rather than release.

6. CONCLUSIONS

The activities of the programmers including copy, paste and modify result in lots of code clone in the software systems. However, Clone group mapping has a very important significance in the evolution of code clones. The clone group mapping approach based on topic modeling is proposed in the paper. By using topic modeling techniques to transform the mapping problem of high-dimensional code space into a low-dimensional topic space, the goal of clone group mapping was indirectly reached by mapping clone group topics. Experiments on 12 versions of 4 open source software show that the recall and precision are up to 0.99, thus the approach can effectively and accurately reach the goal of clone group mapping.

REFERENCES

[1] Bettenburg N, Shang W, Ibrahim W, et al. An Empirical Study on Inconsistent Changes to Code Clones at Release Level[C]//Proc. of the 2009 16th Working Conference on Reverse Engineering. IEEE Press, pp. 85-94, 2009.

[2] Zibran M F, Roy C K. The Road to Software Clone Management: A Survey[R], Technical Report 2012-03, The University of Saskatchewan, Canada, 2012, pp. 1-66.

[3] M. Kim, V. Sazawal, D. Notkin, and G. C. Murphy, "An Empirical Study of Code Clone Genealogies," Proc. ESEC-FSE, 2005, pp. 187–196.

[4] F. Rahman, C. Bird, P. Devanbu, "Clones: What is that Smell?," Proc. MSR, 2010, pp. 72–81.

[5] Bakota T, Ferenc R, Gyimothy T. Clone smells in Software evolution[C]//IEEE International Conference on Software Maintenance. Washington DC: IEEE Computer Society, 2007:24-33.

[6] Barbour L, Khomh F, Zou Y. Late propagation in software clones[C]//Proceedings of the 27th IEEE International Conference on Software Maintenance. Washington DC:IEEE Computer Society, 2011: 273-282.

[7] Gode N, Koschke R. Incremental Clone Detection[C]//Proceedings of the 2009 European Conference on Software Maintenance and Reengineering. Washington DC:IEEE Computer Society, 2009: 219-228.

[8] Saha R K, Roy C K, Schneider K A. An automatic framework for extracting and classifying near-miss clone genealogies[C]//Software Maintenance (ICSM), 2011 27th IEEE International Conference on. IEEE, 2011: 293-302.

[9] Duala-Ekoko E, Robillard M P. Tracking Code Clones in Evolving Software[C]//Proceedings of the 29th international conference on Software Engineering. Washington DC:IEEE Computer Society, 2007:158-167.

[10] C.X. Zhai, Statistical language models for information retrieval, Synthesis Lectures on Human Language Technologies 1 (1) (2008) 1–141.

[11] I. Porteous, D. Newman, A. Ihler, A. Asuncion, P. Smyth, M. Welling, Fast collapsed Gibbs sampling for latent Dirichlet allocation, in: Proceeding of the 14th International Conference on Knowledge Discovery and Data Mining, 2008, pp. 569–577.

[12] D.M. Blei, J.D. Lafferty, Topic models, in: Text Mining: Classification, Clustering, and Applications, Chapman & Hall, London, UK, 2009, pp. 71–94.

[13] Kuhn A, Ducasse S, Gírba T. Semantic clustering: Identifying topics in source code. Information and Software Technology, 2007, 49(3):230–243

[14] Thomas S W, Adams B, Hassan A E, et al. Studying software evolution using topic models[J]. Science of Computer Programming, 2012

[15] Asuncion H, Asuncion A, Taylor R. Software traceability with topic modeling.32nd ACM/IEEE International Conference on Software Engineering (ICSE). 2010:95–104

[16] Tian K, Revelle M, Poshyvanyk D. Using Latent Dirichlet Allocation for automatic categorization of software. 6th IEEE International Working Conference on Mining Software Repositories (MSR). 2009:163–166

[17]Gethers M, Savage T, Di Penta M, et al. CodeTopics: Which topic am I coding now? 33rd International Conference on Software Engineering (ICSE). 2011:1034–1036

[18] HAN Xiaodong ,WANG Xiaobo, LIU Chao.Retrieval method for traceability links between source code and Chinese documentation[J]. Journal of Hefei University of Technology: Natural Science, 2010，33（2）：188-192.

[19] JIN Jing, LI Meng, HUA Zhebang, SONG Huaida, ZHAO Junfeng, XIE Bing. Code function recognition approach based on LDA and static analysis[J]. Computer Engineering and Applications,2013(15).

[20] Liu Y, Poshyvanyk D, Ferenc R, et al. Modeling class cohesion as mixtures of latent topics[C]//Software Maintenance, 2009. ICSM 2009. IEEE International Conference on. IEEE, 2009: 233-242

[21] Lukins S, Kraft N, Etzkorn L. Bug localization using latent Dirichlet allocation. Information and Software Technology, 2010, 52(9):972–990.

AUTHORS

Ruixia Zhang, born in 1989, master,student at Inner Mongolia normal university.Her current research interests include software englneering, code analysis.

Liping Zhang, born in 1974, master, professor at Inner Mongolia normal university.Her current research interests include software englneering, code analysis.

Huan Wang, born in 1991, master,student at Inner Mongolia normal university.His current research interests include software englneering, code analysis.

Zhuo Chen, born in 1989, master,student at Inner Mongolia normal university.His current research interests include software englneering, code analysis.