A Novel Approach of Components Retrieval in Large-scale Component Repositories

Lei Zhang, Lichao Chen, Lihu Pan, Yingjun Zhang
School of Computer Science and Technology
Taiyuan University of Science and Technology
Taiyuan, Shanxi Province, China
chen_lc@263.net

Abstract—**For improving the retrieval performance of large-scale component repositories, a novel approach of components retrieval, Automatic Tags Extraction(ATE) retrieval, is proposed in this paper. In this method, component tags are extracted automatically from application domain terms, high-frequency terms, high-weight terms and facet terms in description document of component at first, and then the improved VSM (Vector Space Mode) similarity algorithm is used to retrieve on the tags. Our experiments show that, compared with some common retrieval methods, the ATE retrieval is more feasible and efficient.**

Keywords- software component, component retrieval, automatic tags extraction, web services

I. INTRODUCTION

The application of component-based technique can substantially improve the productivity and reliability[1] of software development. The idea of software reuse has also been well represented in some applications of software services such as Web Services and Active Services[2]. Higher requirements for service efficiency are propounded by the development of the service-oriented techniques. It is dependent upon the capability of the component management and retrieval system in the repository to provide the needed components for the services quickly and accurately. With the scaling up of the component repository, the requirements of the dynamic-demand services can not be met efficiently by the traditional components retrieval approaches. How to make components retrieval maintaining a high recall and precision ratio while the retrieval time decreasing substantially, it is still an open question that should be solved quickly.

An Automatic Tags Extraction(ATE) algorithm is designed for components retrieval in this paper. With the ATE algorithm, a novel approach of components retrieval -- ATE retrieval is proposed. Firstly, component tags are extracted and indexed by the ATE algorithm, and then an improved VSM similarity algorithm is used to match the component tags. The application of ATE algorithm in description documents of components will find more useful information and significantly improve the recall ratio; the precision ratio is also raised by applying the similarity algorithm. Since applying index mechanism, the ATE retrieval can achieve components retrieval with a fast speed.

978-1-4673-2008-5/12/$31.00 ©2012 IEEE

This research was supported by The Natural Science Foundation of Shanxi Province of China under Grant No.2009011022-1, and Graduate Innovative Projects of Taiyuan University of Science and Technology Grant No.20111025.

II. THE PROCESS OF ATE RETRIEVAL

Faceted classification scheme is a widely used method for components classification and the ATE retrieval is base on the faceted classification scheme too. Since XML is very suitable for describing the tree structure and provides sufficient support for adding new applications, component information is stored in XML documents. The component description documents which contain more information than the faceted classification are stored as a child node in the XML documents.

The process of components retrieval is shown in Fig.1.

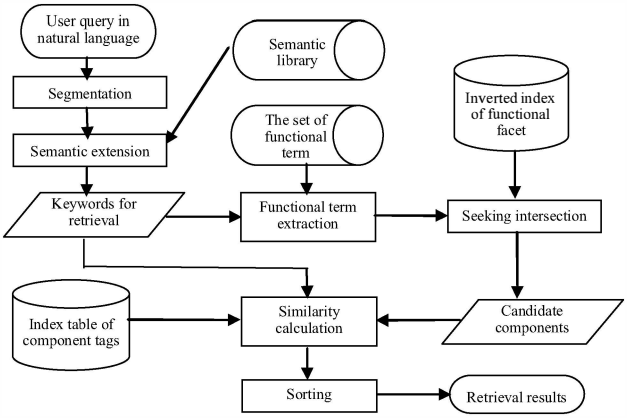

Figure 1. ATE retrieval process

1) Queries input by the user

Specifying the queries in basically unrestricted natural language does not need extra learning costs, and can improve the efficiency of components retrieval.

2) Retrieval sentence segment and semantic expand

The semantic expansion stage uses the generalized relationship, synonymous relationship to expand the segmented results and to gain the retrieval keywords. With the semantic expansion, the latent semantics of users in retrieval are gained.

3) Functional term extracted in keywords

According to the set of functional terms, the functional terms in retrieval keywords are extracted for the next step.

4) Seek intersection of functional term

The components which meet all the functional requirements are selected as candidate set by seeking in the inverted index of functional facet. This index consists of two parts: each term in functional facet as an index entry and every component contained that term as the corresponding index queue. The index queues are ascending sorted by component ID for seeking intersection. The components which are not meeting the right functional requirements can be quickly excluded by using this index, and retrieval time is reduced significantly.

5) Similarity calculated

For each component in the candidate set, the tag information is read from the tag index, and then is matched with the retrieval keywords using improved VSM similarity.

6) Retrieval results sorted according to the similarity.

III. AUTOMATIC TAGS EXTRACTION

A. Tags description

Tags should contain the most information of the component. Considering the factors such as frequency and weight, and combining with the characteristics of component retrieval, the formula of tags extracting is designed as (1). Where DT is the set of terms that belong to the component application domains, the ATC is the set of terms which are contained in the component description document, the HWT is the set of high-weight terms, the HFT is the set of high-frequency terms and the FT is the set of faceted terms of the component. The ATC and FT set can read from XML document directly, the DT, HFT, HWT set is described as fallow.

$$\text{Tags}=\{[(\text{DT}\cap\text{ATC})\cup\text{HWT}]\text{-HFT}\}\cup\text{FT} \quad (1)$$

- DT set. Despite may having little weight in the component, DT terms have a good differentiating ability, so these terms should be set as tags. Since the DT frequently appear in the relevant domain, so that it have more average weight in corresponding domains than other domains. From the characteristic mentioned above, the DT extracting formula is designed as (2). Where T_d is the domain factor in domain D, W_t is the weight of the term in component, N_d is the number of components belonged to domain D, N is the number of all components. When T_d is greater than the threshold, the corresponding term are considered as DT. After comparing and experimenting repeatly, the threshold of DT is set at 1.7.

$$T_d = \frac{\sum_{c\in D} W_t}{N_d} \Big/ \frac{0.01+\sum_{c\notin D} W_t}{N-N_d} \quad (2)$$

- HWT set. The term weight is calculated using (3), where tf is term frequency, n_t is the number of documents which contain the corresponding term and N is the number of total components. The term is added to HWT when its weight is greater than a threshold. The threshold is set at 70% as default, and can be reset according to the requirement of retrieval time and precision.

$$W = tf \times idf = tf \times \left[1+\log_2\left(\frac{N}{n_t+1}\right)\right] \quad (3)$$

- HFT set. The so-called high-frequency terms are the terms which appeared in so many components. The HFT having lower differentiating ability, are mainly general terms such as "software", "can" and so on. These terms are not suitable for tagging the components. The component frequency F is calculated using (4), and the n_t, N has the same meaning with HWT. The word can be regarded as high-frequency term when the component frequency F of this word is greater than a threshold. Also through comparisons and experiments, the threshold of HWT is set at 0.35.

$$F=n_t/N \quad (4)$$

B. The ATE algorithm

Algorithm 1 The ATE algorithm

```
create union term-vector
initialize domain weight list
for each component do
  calculate FT set
  for each term in the component do
    add to ATC set
    create the T-C(Term-Component) matrix
    add to the right domain-weight list
  end for
end for
for each term in union term-vector do
  calculate F use formula (4)
  if F> threshold f then add this term to HF
  for each domain do
    calculate T_d using formula (2)
    if T_d> threshold t_d then add to the DT set of corresponding domain
  end for
end for
for each component do
  extracte HW from T-C matrix
  use formula (1) to calculate the tags of this component
  index tags
end for
```

IV. IMPROVED VSM SIMILARITY FORMULA

When matching the retrieval keywords with the component tags, the similarity formula used in this paper is based on VSM. With the simplifying of VSM similarity formula and adding some factors such as *FacetBoost* and *Hits* for component retrieval, the precision of component retrieval can be increased by using the improved similarity formula.

The retrieval keywords and component tags are identified in vector space as weight vector V_q and V_c. The VSM similarity is represented using the cosine of the angle between the vectors[5] as follow:

$$sim(q,c)=\frac{\sum_{i=1}^{n}\left(tf\left(t_i,q\right)\times idf\left(t_i,q\right)\times tf\left(t_i,c\right)\times idf\left(t_i,c\right)\right)}{\sqrt{\sum_{tinq}\left(tf\left(t,q\right)\times idf\left(t,q\right)\right)^2}\sqrt{\sum_{tinC}\left(tf\left(t,c\right)\times idf\left(t,c\right)\right)^2}} \quad (5)$$

Simplifying the VSM similarity formula according to the following characteristics of component retrieval:

- Only when $t_i \in T_q \cap T_c$, the corresponding term of inner product can have a non-zero value, where the T_q is the set of keywords and the T_c is the set of component tag. Because the T_q is generally smaller than T_c, so that only the term of $ti \in T_q$ are calculated for reducing computational cost.
- Since rarely containning the same terms in query sentence, it can be assumed that $tf(ti,q)=1$.
- A term, whether appeared in the retrieval keywords or in the tags, has the same *idf*, that is, $idf(ti,q)=idf(ti,c)=idf(ti)$.
- Since all ranked components are multiplied by a same factor, the factor $\sqrt{\sum_{tinq}\left(tf\left(t,q\right)\times idf\left(t,q\right)\right)^2}$ does not affect component ranking, so that it is not considered in the paper.

The simplified similarity formula is represented as follow:

$$sim\left(q,c\right)=\frac{\sum_{tinq}\left(tf\left(t,c\right)\times idf\left(t\right)^2\right)}{\sqrt{\sum_{tinc}\left(tf\left(t,c\right)\times idf\left(t\right)\right)^2}} \quad (6)$$

The *FacetBoost* and *Hits* are adding to (6) for component retrieval:

- *FacetBoost(t)*: *FacetBoost(t)* is a boost of faceted term *t* when *t* is hit. For improving the differentiation of term, different facet has different *FacetBoost* when a term is hit. For example, the *function* and *quality of service* facet can be set greater boost than other facet when higher requirements are needed in function and QoS. These factors can be set by user for various requirements of retrieval.
- *Hits*: the components can be further rewarded through *Hits* when more keywords are matched. The *Hits* is calculated using $Hits=n_h/n_q$, where n_h is matched number and n_q is the number of all retrieval keywords.

The final similarity formula is represented as (7):

$$sim\left(q,c\right)=Hits\times\frac{\sum_{tinq}\left(tf\left(t,c\right)\times idf\left(t\right)^2\times FacetBoost(t)\right)}{\sqrt{\sum_{tinc}\left(tf\left(t,c\right)\times idf\left(t\right)\right)^2}} \quad (7)$$

V. VALIDATION

A. Preparation for the experiments

A prototype component repository is built for the experiments with 2547 components distributed in 14 application domains from Shanghai component library[6]. These domains include administrative office, finance, commerce, software development, and so on. Some other widely used retrieval approaches[7][8] are compared with our approach under the same retrieval conditions.

The experiments are divided into four groups using different approaches respectively, group A using completely inverted index, group B using full-text retrieval, group C using facet tree matching and group D using the ATE retrieval. Each group consists of seven members and conducts retrieval thirty times in same retrieval platform. The retrieval platform is Intel Pentium T4200， 2GDDR2 RAM， 320G hard disk with Windows Vista，Myeclipse7.0，and jdk1.6.0-17 environment. The same modules are used in the segmentation and semantic extension stage for each group. The segmentation tool is the IKAnalyzer3.2.5stable. Since loading the dictionary for Chinese segmentation in first segmenting, about 800 ~ 900 ms are included in the retrieval time in each group.

Four metrics are considered in the experiment: recall, precision, f-measure, retrieval time. Recall is the number of relevant components retrieved over the number of relevant components in the repository. The precision is the number of relevant components retrieved over the total number of components retrieved. Recall and precision are the classic measures of the effectiveness of an information retrieval system. The f-measure is the harmonic mean of precision and recall. The closer the f-measure is to 1.0, the better the retrieval approach is. F-measure is calculated using $2PR / (P + R)$ where the P is precision and R is recall. The retrieval time is the time in narrow sense, refers to the time from the user input query sentence to the results being given. The retrieval time is expressed in seconds.

B. Analysis of Experimental Results

The retrieval effectiveness is shown in Fig.2. Group A with completely inverted index has the highest precision ratio, but the lowest recall ratio since this approach can not matching partly; Group B with full-text retrieval has the highest recall ratio, but lowest precision ratio; Group C with facet tree matching and group D with the ATE retrieval has the better f-measure than group A and B.

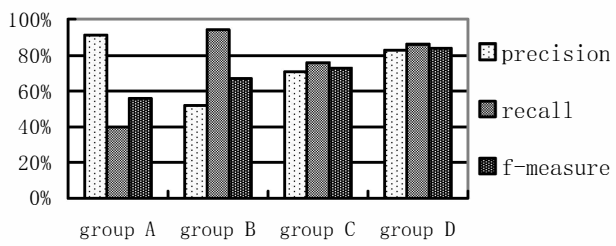

Figure 2. Contrast figures of retrieval effectiveness

The retrieval time is shown in table 1. It can be seen that group D has the best retrieval time among four groups while keep better retrieval effectiveness. Group C though has better retrieval effectiveness than group A and B, but the retrieval time is still longer than group D.

In order to validate the effectiveness of ATE retrieval in large-scale component repositories, some simulated repositories are built using random selecting algorithm. The components in prototype repository are selected evenly by the algorithm and stored in the simulated repositories. The retrieval time of ATE in different size repositories is shown in Fig.3. It can be seen that the increasing of retrieval time is less than the logarithm level.

TABLE I. RTRIEVAL TIME OF EACH GROUP

Group	Group A	Group B	Group C	Group D
Retrieval time(Seconds)	1.054	8.761	6.076	0.968

The experiment results show that the ATE retrieval can greatly decrease retrieval time while keeping a high level recall ratio. Since using the improved VSM similarity for component retrieval, the precision is also at a high level. The ATE retrieval, having a better retrieval effectiveness and faster retrieval time, is very suitable for the applications of large-scale component repository retrieval.

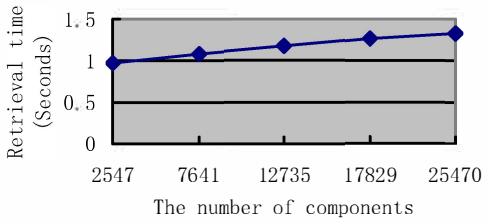

Figure 3. Increasing trends of retrieval time

VI. CONCLUSIONS

With the rapid development of the componentized software services, higher requirements for service efficiency are propounded, and as the constant expansion of component repository these requirements are more and more urgent. A novel approach--ATE retrieval is proposed in this paper to improve the effectiveness of component retrieval in large-scale repository. This approach first uses the ATE algorithm to extract the tags of components, and then pre-calculate the information such as weight of tags and index this information to decrease retrieval time. Experiments show that ATE can reduce the noise information in the described document of component effectively and significantly decrease the retrieval time. Since being able to keep a good balance between retrieval effectiveness and time by adjusting thresholds of ATE, the ATE retrieval is very suitable for the retrieval applications of large-scale or time-critical component repositories.

Future work includes two directions. On the one hand, we will research how to improve the similarity matching algorithm to further increase precision ratio. On the other hand, we will research how to improve the effectiveness of component retrieval in distributed isomeric reuse repositories.

[1] Ivar J. Software reuse: Architecture, process and organization for business success. Reading: Addison-Wesley Publishing Company, 1997. 4-15.

[2] Jian Yang. Web Services Componentization. Communications of the ACM, 2003, 46(10): 35-40.

[3] Prieto-Diaz R. Implementing faceted classification for software reuse. Communications of the ACM, 1991, 34(5):88-97.

[4] Ruben Prieto-Diaz, Peter Freeman. Classifying software for reusability. IEEE Software, 1987, 4(1): 6-16.

[5] Salton G. Developments in automatic text retrieval. Science, 1991, 253(5023):974-979.

[6] Shanghai component library[DB/OL]. [2011-04-05]. http://www.sstc.org.cn/.

[7] Xu ru zhi, Qian le qiu, Cheng jian ping, Wang yuan feng, Zhu san yuan.Research on Matching Algorithm for XML-Based Software Component Query. Journal of software, 2003, 14(7):1195-1202.

[8] Zhong Ming. Research on Component Universal Description and Retrieval in Active Services [Ph.D. Thesis]. Beijing:Tsinghua University, 2010.

[9] Awny Alnusair, Tian Zhao. Component Search and Reuse: An Ontology-based Approach. In: Proc.of the IEEE International Conference on Information Reuse and Integration (IRI), 2010, 258 -261.

[10] Rajesh K. Bhatia, Mayank Dave, R. C. Joshi. Ant Colony Based Rule Generation for Reusable Software Component Retrieval. ACM SIGSOFT Software Engineering Notes, 2010, 35(2):1-5.

[11] Stephen S. Yau, Junwei Liu. Service Functionality Indexing and Matching for Service-Based Systems. IEEE International Conference on Services Computing, 2008, 461-468.

[12] Jasmine Kalathipparambil Sudhakaran, Ramaswamy Vasantha. A Mixed Method Approach for Efficient Component Retrieval from a Component Repository. Journal of Software Engineering and Applications, 2011,4:442-445.