Journal of Software Engineering and Applications, 2011, 4, 442-445
doi:10.4236/jsea.2011.47051 Published Online July 2011 (http://www.SciRP.org/journal/jsea)

A Mixed Method Approach for Efficient Component Retrieval from a Component Repository

Jasmine Kalathipparambil Sudhakaran[1], Ramaswamy Vasantha[2]

[1]Rashtreeya Vidyala College of Engineering, Bangalore, India; [2]Department of Information Science and Engineering, Rashtreeya Vidyala College of Engineering, Bangalore, India.
E-mail: jasminesadeep@yahoo.co.in

Received April 21st, 2011; revised May 19th, 2011; accepted May 28th, 2011.

ABSTRACT

A continuing challenge for software designers is to develop efficient and cost-effective software implementations. Many see software reuse as a potential solution; however, the cost of reuse tends to outweigh the potential benefits. The costs of software reuse include establishing and maintaining a library of reusable components, searching for applicable components to be reused in a design, as well as adapting components toward a proper implementation. In this context, a new method is suggested here for component classification and retrieval which consists of K-nearest Neighbor (KNN) algorithm and Vector space Model Approach. We found that this new approach gives a higher accuracy and precision in component selection and retrieval process compared to the existing formal approaches.

Keywords: *Software Reuse, Component Retrieval, Vector Space Model Algorithm*

1. Introduction

Many software organizations realized that developing the software using reusable components could dramatically reduce development effort, cost and accelerate delivery. But the non-existence of a standard searching technique for finding the suitable component and also the lack of appropriate tool in this field contributed towards in large-scale failures in their approach. From the past studies on this field, it is found that researchers are tried with different approaches to improve the adaptability of the component but very few studied had taken place in improving the efficiency of component retrieval. Fuzzy linguistic approach is familiar in the information retrieval process [1]. In this paper we have used an algebraic model namely Vector Space model in which text documents are represented as vectors of identifiers, such as, index terms which is used in information filtering, information retrieval, indexing and relevancy rankings along with K-Nearest Neighbor(KNN) algorithm for classification of documents. The paper is structured as follows. Section 1 starts with a discussion of what is meant by software reuse and a reusable component. Section 2 talks about present scenario in the component retrieval process. Section 3 presents methodology adopted. Section 4 mentions some concluding remarks and the relevance of related models which can be extended from the vector space model and various combination algorithms which can contribute towards further extension of this work are pointed out.

1.1. What Is Software Reuse?

Software reuse at its most basic level consists of making use of any existing information, component or product when designing and implementing a new system or product.

There are differing opinions as to which activities constitute genuine software reuse. Replication of an entire software program does not count as reuse. Reuse of assets is dependent upon both similarities and differences between the applications in which the component is being used [2].

Many organizations already practice a limited form of reuse, for example, most developers have libraries of components that they have developed in previous projects, or they use standard libraries, which are available with many programming languages [1]. About 30% of the cases, it is a very ad-hoc method of reuse, and it will work very well on a small scale and it will not be suitable for entire organizations [3]. Instead, businesses need to implement a systematic reuse program in order to gain the full advantages of reuse.

1.2. What Can Be Reused?

The definition of a reusable component is “any component that is specifically developed to be used, and is actually used, in more than one context” [3]. This does not just include code; other products from the system lifecycle can also be reused, such as specifications and designs, and even requirements on occasion [4]. ‘Components’ in this case can be taken to include all potentially reusable products of the system lifecycle, including code, documentation, design, requirements etc.

There are various criteria that should be satisfied in order for an asset to be successfully reusable. These are grouped into General, Functional and Technical requirements [5]. General requirements focus on aspects such as compliance with relevant standards, completeness, modularity and simplicity. All components should conform to the General requirements. Functional requirements include such concerns as which business processes it will simulate or automate, and how well it does this. Functional requirements mainly concern Vertical or Domain-specific assets and tend to be very specific to each information domain. Lastly, Technical requirements refer to criteria such as interoperability, portability, communication, security etc [2].

There are different levels of reuse, which can be considered [3]. At the highest level, entire applications can be reused on different platforms provided they are portable. Sub-systems can be reused within different applications, possibly within different domains. Reusable assets can be also being built in-house, retrieved from legacy systems or can be bought from an external source.

2. Present Scenario in the Component Retrieval Process

Existing approaches to software component retrieval process cover a wide spectrum of component encoding methods and search or matching algorithms. The encoding methods differ with respect to their soundness, completeness, and the extent to which they support an estimate of the effort it takes to modify a component. Text-based encoding and retrieval is neither sound nor complete. Its disadvantages have been thoroughly in the information retrieval literature [5,6]. Lexical descriptor-based encoding approach also suffers from a number of problems about developing and using classification vocabulary [7]. Software specific challenges include the fact that one-word or one-phrase abstractions are hard to come by in the software domain [8]. From the user’s point of view, lack of familiarity with the vocabulary is also pointed out as draw back in using a component retrieval system effectively [9]. In this context Vector space Model will be a promising solution for component retrieval process [10,11].

3. Methods Used

3.1. Vector Space Model Approach

It is an algebraic model in which documents and queries are represented as vectors [12] as follows:

dj = (w1,j,w2,j,···,wt,j)

q = (w1,q,w2,q,···,wt,q)

Each dimension corresponds to a separate term. If a term occurs in the document, its value in the vector is non-zero. An indexed collection of documents is represented as a term table which has documents as fields and words as primary key for row. The (D)i(Word)j-th entry of this table records how many times the j-th search term appeared in the i-th document. The following **Figure 1** shows a sample vector space model.

The first major component of a vector space search model is the concept of a term space. A term space consists of every unique word that appears in a collection of documents. The second major component of a vector space search model is term counts. Term counts are simply records of how many times each term occurs in an individual document. This is represented as a table. By using the term space as a coordinate space, and the term counts as coordinates within that space, we can create a vector for each document. As the number of terms increases, the dimensionality of VSM also increases. **Figure 2** shows the structure of the word database and **Figure 3** shows the structure of the rank table.

For these words documents and corresponding ranks will be stored in the rank table.

Based on the ranking terms are compared as “ranked higher than”, “ranked lower than” or “ranked equal to” the second, making it possible to evaluate complex information according to query criteria. Here search vector space

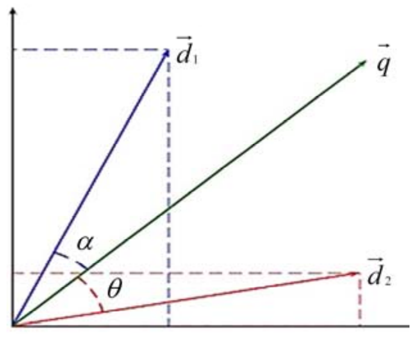

Figure 1. A sample vector space model.

Words	d_1	d_2	d_3	d_4	…	d_n
	1	1	1	1	0	1
	3	0	2	1	0	1
	0	1	3	0	0	2
	0	0	0	0	0	0

Figure 2. Structure of word database.

Dno	Rank
d_1	1
d_2	3
…	…
d_n	1

Figure 3. Structure of rank table.

search model ranks the documents it finds according to the estimation of their relevance, making it possible for the user quickly to select the components according to their requirements [7,13].

Relevancy rankings of documents in a keyword search can be calculated, using the assumptions of document similarities theory, by comparing the deviation of angles between each document vector and the original query vector where the query is represented as same kind of vector as the documents.

It is easier to calculate the cosine of the angle between the vectors instead of the angle:

$$\cos\theta = \frac{d_2 \cdot q}{\|d_2\|\|q\|}$$

A cosine value of zero means that the query and document vector are orthogonal and have no match (*i.e.* the query term does not exist in the document being considered).

3.2. The Document Classification Algorithm Employed

KNN classifier is an instance-based learning algorithm that is based on a distance function for pairs of observations, such as the Euclidean distance or Cosine. The k-Nearest Neighbour (kNN) classifier algorithm has been studied extensively for text categorization by Yang and Liu [6]. In this classification paradigm, k nearest neighbors of a training data are computed first. Then the similarities of one sample from testing data to the k nearest neighbors are aggregated according to the class of the neighbors, and the testing sample is assigned to the most similar class. The similarity in score of each neighbour document to the test document is used as the weight of the categories of the neighbour document [8]. If there are several training documents in the k nearest neighbour, which share a category, the category gets a higher weight. In this work, we used the Cosine distance to calculate the similarity score for the document representation.

One of advantages of KNN is that it is well suited for multi-modal classes as its classification decision is based on a small neighborhood of similar objects (*i.e.*, the major class). So, even if the target class is multi-modal (*i.e.*, consists of objects whose independent variables have different characteristics for different subsets), it can still lead to good accuracy. A major drawback of the similarity measure used in KNN is that it uses all features equally in computing similarities. This can lead to poor similarity measures and classification errors, when only a small subset of the features is useful for classification [5].

3.2.1. Steps for KNN Using Average Cosine

Step 1: Select k nearest training documents, where the similarity is measured by the cosine between a given testing document and a training document.

Step 2: Using cosine values of k nearest neighbors and frequency of documents of each class i in k nearest neighbors, compute average cosine value for each class i, Avg_Cosine (i).

Step 3: Classify the testing document a class label which has largest average cosine.

In order to reduce the dimensionality of VSM and keep useful information, we first compute concept vectors for given categories. Then, using the concept vectors as projection matrix, projection of both training and testing data is done. Finally, we apply KNN algorithm on the projected VSM model that has reduced dimensionality.

3.2.2. Steps of Combined Approach for Vector Based Algorithm and K-Nearest Neighbor Algorithm

Step 1: Compute a concept vector for each category using true label information of training documents and then construct concept vector matrix C (w-by-c), where c is the number of categories.

Step 2: Do projection of VSM model A (w-by-d) using concept vector matrix C (w-by-c) (*i.e.*, C^T * A).

Step 3: Apply KNN with the projected VSM model (*i.e.*, c-by-d matrix).

4. Conclusions and Future Work

A novel KNN classification algorithm combining model and evidence theory is proposed in this paper. The new method not only overcomes the main shortage of lazy learning in traditional KNN, but also takes the distances between samples to be recognized and samples in k-neighbors into account. At the same time the method resolves the unrecognizable cases of unknown samples. Applying the classification algorithm into the document recognition, experimental results show its satisfied recognition rate and fast categorization speed.

There are also models based on and extending the vector space model such as generalized vector space model, Topic-based Vector Space Model and latent semantic indexing etc and also combination algorithms which consists of clustering, Singular Value Decomposition(SVD)-based Algorithm, Naive Bayesian Algorithm and variations of KNN algorithm. Future work can be aimed in these directions.

REFERENCES

[1] W. S. Sarma and V. Rao, "A Rough–Fuzzy Approach for Retrieval off Candidate Components for Software Reuse," *Pattern Recognition Letters*, Vol. 24, No. 6, 2003, pp. 875-886. doi:10.1016/S0167-8655(02)00199-X

[2] P. A. González-Calero, "Applying Knowledge Modeling and Case-Based Reasoning to Software Reuse," *IEE Proceedings – Software*, Vol. 147, No. 5, October 2000, pp. 169-177.

[3] D. Lucrédio, *et al*., "Component Retrieval Using Metric Indexing," *IEEE International Conference on Information Reuse and Integration*, Las Vegas, 8-10 November 2004, pp. 79-84.

[4] D. Lucrédio, *et al*., "A Survey on Software Components Search and Retrieval," 30*th IEEE Euromicro Conference*, Rennes, 31 August-3 September 2004, pp. 152-159.

[5] G. Salton, A. Wong and C. S. Yang, "A Vector Space Model for Automatic Indexing," *Communications of the ACM*, Vol. 18, No. 11, 1975, pp. 613-620. doi:10.1145/361219.361220

[6] L. S. Sorumgard, G. Sindre and F. Stokke, "Experiences from Application of a Faceted Classification Scheme," *Advances in Software Reuse*, *Selected Papers from the* 2*nd International Workshop on Software Reusability*, Lucca, 24-26 March 1993, pp. 116-124.

[7] J. B. Lovins, "Development of a Stemming Algorithm," *Mechanical Translation and Computational Linguistics*, Vol. 11, No. 1-2, 1968, pp. 22-31.

[8] D. Blair and M. E. Maron, "An Evaluation of Retrieval Effectiveness for a Full-Text, Document-Retrieval System," *Communications of the ACM*, Vol. 35, No. 4, March 1985, pp. 289-299. doi:10.1145/3166.3197

[9] Y. Yang and X. Liu, "A Re-examination of Text Categorization Methods," *Proceedings of ACM SIGIR Conference on Research and Development in Information Retrieval*, Berkley, 15-19 August 1999, pp. 42-49.

[10] W. B. Frakes and B. A. Nejmeh, "An Information System for Software Reuse, Software Reuse: Emerging Technology," CS Press, Sheffield, 1990, pp.142-151.

[11] Y. Yang and J. O. Pedersen, "A Comparative Study on Feature Selection in Text Categorization," *Proceedings of the* 14*th International Conference on Machine Learning*, Nashville, 8-12 July 1997, pp. 412-420.

[12] N. Ishii, T. Murai, *et al.*, "Text Classification by Combining Grouping, LSA and kNN," 5*th IEEE/ACIS International Conference on Computer and Information Science*, Honolulu, 10-12 July 2006, pp. 148-154.

[13] Y. S. Yoelle S. Maarek, D. M. Berry and G. E. Kaiser, "An Information Retrieval Approaches for Automatically Constructing Software Libraries," *IEEE Transactions on Software Engineering*, Vol. 17, No. 8, 1991, pp. 800-813. doi:10.1109/32.83915