Input Data Representation for Self-Organizing Map in Software Classification

Yuqing Lin and Huilin Ye
School of Electrical Engineering & Computer Science 1 The University of Newcastle
Callaghan, Australia
{Yuqing.Lin, Huilin.Ye}@newcastle.edu.au

Abstract—A self-organizing map (SOM) is used to classify software documents and the associated software components with the aim to facilitate software reuse. SOM learns from input stimuli rather than training data, therefore the quality of input data representation is crucial to the success of SOM. In this paper, we use automatic indexing method to represent a document collection as the input data to train a SOM. The automatic indexing uses a phrase formation method to promote precision and a domain dependent relational thesaurus to enhance recall. A retrieval experiment based on a document collection containing 97 Unix manual pages was conducted to evaluate the effectiveness of this input data representation scheme. Promising retrieval results were observed.

Keywords-Self-Organising Map; automatic indexing, software classification

I. INTRODUCTION

A number of studies have been devoted to the use of Kohonen self-organizing map (SOM) [1] for classifying textual documents. Kohonen SOM is an unsupervised neural network. No training data is needed in its learning process. Instead, it learns from input data. After learning, input data with similar features will be mapped to adjacent areas of the grid. It is this property of SOM that makes it useful for classification. This is somewhat similar to the human brain in that different regions in human brain are dedicated to specific tasks and the internal representations of information in the brain are generally organized spatially [2].

In this research, a SOM for classification of reusable software assets, called Software Self-Organizing Map (SSOM), is developed. Software classification and retrieval still remain a major challenge to the wide spread adoption of software reuse, although this subject has been investigated for over two decades. The proposed SSOM classifies the software documents rather than the software assets themselves. Most software modules consist of textual descriptions of their contents and functionality. For example, design specifications and software code documentation such as manual pages are generally available in electronic form. These natural-language documents are a rich source of conceptual information. However, the concept information in these documents is contained implicitly, in an unstructured way [3], and is not in a format suitable for input in the SOM. As SOM learns from input stimuli rather than training data, the performance of SOM depends significantly on the quality of the input data. Thus how to appropriately represent input data for the SSOM is a major issue. This paper is focused on this topic. The complete design issues on the SSOM may be found in [4].

The purpose of this research is to find a cost-effective representation scheme for the SSOM input data. An automatic indexing scheme for SOM input data representation is used to represent document collection. Automatic indexing will generate several indices for each document in a collection. Each index denotes a certain feature of the document in which it is extracted. All the indices in the collection will form a feature space. As this set of features will serves as the input data of the SOM, it is called the SOM’s input feature space.

The reminder of this paper is organized as follows. The automatic indexing approach will be introduced in Section 2. The SOM based classification is presented in Section 3. The feasibility and effectiveness of the representation scheme are tested through a retrieval experiment. This will be reported in Section 4. Finally, conclusions will be presented in Section 5.

II. TYPE AUTOMATIC INDEXING

The major tasks involved in indexing are single term identification, phrase formation, and relational thesaurus construction. In addition to single content identifiers, phrases that carry more specific semantic meanings than single terms are needed to promote retrieval precision, and a domain relational thesaurus that groups semantically related terms into term classes is created to enhance retrieval recall. Recall and precision are the most widely used measures for evaluating retrieval effectiveness [5]. Recall is the proportion of relevant material retrieved, whereas precision is the proportion of retrieved material that is relevant.

A. Single Term Identification

Single term identification is relatively simple. After removing all the common function words and stemming in a collection of documents called the corpus, the best single content identifiers are those (1) occurring frequently in individual documents, and (2) occurring not very frequently in the remainder of the collection. To measure the ability of an indexing term to uniquely distinguish a document in a corpus, Salton [6] introduced a parameter \( w_{ij} \), called the weight of a term \( T_j \) in document \( D_i \).

\[
 w_{ij} = tf_{ij} \cdot idf_{ij} = tf_{ij} \cdot \log \frac{n}{df_{ij}}, 
\]

(1)

Here \( tf_{ij} \) is the term frequency for term \( T_j \) occurring in document \( D_i \), \( df_{ij} \) is the document frequency defined as the number of documents in an n-document
collection containing \( T_j \), and \( idf_j \) is the inverse document frequency which indicates how rarely a term \( T_j \) occurs in the remainder of the collection and is given by \( \log(n/df_j) \).

A threshold weight, \( w \), is then chosen. For each document \( D_i \), all terms \( T_j \) for which \( w_j > w \) are selected as single content identifiers associated with \( D_i \).

### B. Phrase Formation

Single terms, used out of context, may carry ambiguous meanings, whereas phrases in general are more specific to permit reasonably unambiguous interpretations. After single descriptors have been identified, it is necessary to go further by forming phrases. Term phrases consisting of sequences of related words carry a more specific meaning than the single terms separately.

Salton [6] stated that term phrases should not be randomly constructed. He proposed a discrimination model which suggests a relationship between the discrimination power and the document frequency. Document frequency indicates how often a term occurs in a document collection. Good discriminators have a medium document frequency, whereas poor discriminators have a high document frequency. Only those terms of higher document frequency can be used to form phrases, which transforms higher document frequency poor discriminating single terms into medium document frequency phrases with greater discriminating power. Phrases that include only low document frequency components will receive a still lower document frequency.

As a result, their value for indexing purpose will be even smaller than that of the single-term components they replace. Words with high document frequency are used as a phrase head to form phrase(s) with the other words, called phrase-component, depending on the distance between phrase-head and phrase-component. The distance is measured by the number of words occurred between phrase-head and phrase-component in the source document, and the context in which phrase-head and phrase-component co-occur in the original document.

Phrase-head and phrase-component must be adjacent words after removing stop words, and should not span any boundaries of sentences, sections and paragraphs. A phrase \( P \) thus can be described as a binary relation

\[
P(p_1, p_2)
\]

where \( p_1, p_2 \) are elements of phrase \( P \). At least one of the elements must meet the criterion for phrase-head. The following measure of phrase weight is defined:

\[
w_p = \frac{w_1 + w_2}{2}
\]

where \( w_p \) is the weight of phrase \( P \), \( w_1 \) is the weight of \( p_1 \), and \( w_2 \) is the weight of \( p_2 \). This formula was chosen for two reasons. Firstly, since the phrase weight is a function of the weights of the phrase elements, it incorporates information about the importance of the elements of the phrase into the phrase weight. Secondly, it assures that the magnitude of phrase weight does not differ greatly from the magnitude of single term weight.

Setting a threshold of phrase weight. Phrases with a weight greater than the threshold will be selected as indices.

### C. Relational Thesaurus Construction

At the stage of retrieval of reusable software assets, in addition to exact matches, it is desirable that semantically relevant or closely related matches are also recovered, so to provide the user with a variety of reuse alternatives to choose from and modify.

Relational thesauri have been shown to be useful in increasing recall from information retrieval systems. A thesaurus is a grouping of words, or word stems, into certain subject categories, called concept classes. The terms in the thesaurus classes can replace the initial search terms and can be used to increase retrieval recall. For example, if a document is relevant for a certain query, but uses indexing terms synonymous to the words expressed in the query, it would normally not be retrieved. One solution is to expand the query through synonym relations as found in a thesaurus [7]. The concept classes may be different in different domains. Generic thesauri of this sort will often not be specific enough for the text collection at hand [7]. A domain dependent thesaurus is necessary for identification of the relationships between terms contained in the text collection.

The method used in this research is semiautomatic. First, use a program to automatically build a term-document matrix and calculate the term relatedness coefficient based on the term co-occurrence in the collection. Next, a threshold of term relatedness coefficient will be determined, and then a set of term pairs whose term relatedness coefficients are greater than the threshold will be identified. Finally, the actual relationships between the term pairs will be recognized by human. Thus involvement of human effort is limited to a minimum as the number of term pairs has been significantly reduced after the selection.

The identified relationships among the terms will be stored in a relational thesaurus, which will be used in both the classification and retrieval processes.

Based on the method described above, a domain dependent relational thesaurus was constructed for a document collection including 97 UNIX command manual pages. The relatedness coefficients were calculated for all term pairs contained in the collection. The threshold of relatedness coefficient was set to 3. Several hundreds of term pairs with a coefficient greater than the threshold were found to be eligible for entries into the relational thesaurus. Three kinds of relationships were identified in this domain:

- Synonyms (S)
- Generic-Specific (G-S)
- Whole-Part (W-P)

Synonym relationship is used in both classification and retrieval stages. If term 1 is defined as being equivalent to term 2, they are considered as synonyms, such as \( \text{fil} \rightarrow \text{filename} \). In the domain of UNIX commands, \( \text{filename} \), as a parameter of a command, usually indicates a \( \text{file} \). All the synonyms will be merged into one single descriptor class at the stage of classification. In addition, the single descriptor class is used in the retrieval stage rather than the individual terms belonging to the class.

The G-S relationship represents a hierarchical structure. This makes it possible to classify more specific topics under more general ones and to formulate a search request by starting with a general formulation and
progressively narrow the specification down to those areas that appear to be of principal interest [6].

The W-P relationship includes two sub-relationships, attributive and constituent which are defined by [8].

- **Attributive**: term 2 is defined with respect to one or more distinctive or characteristic attributes of term 1. The relationship between “file” and its “name” is an attributive relationship.

- **Constituent**: term 2 is defined as being a constituent or part of term 1. The relationship between “file” and “page” is a constituent relationship.

The Modifier-Modified relationship actually stands for a phrase. There is no distance or context constraint put on the two terms. The generated M-M relationships, or phrases, are based on the co-occurrence in certain documents, rather than the adjacency of the phrase elements. Examples such as print-fil, copy-fil, display-fil, link-fil, creat-fil are very good descriptors in terms of the functionality of UNIX command. These M-M pairs will be collected into the phrase thesaurus as a supplement of the functionality of UNIX command.

These M-M pairs will be elements. Examples such as file documents, rather than the adjacency of the phrase elements.

The output layer is a two-dimensional grid, each cell of which is associated with a weight vector whose dimension equals to the number of the identified features. If a feature is assigned to the document, the correspondent element of the vector will be assigned a value equaling to the weight of the feature, otherwise 0 will be assigned to the element. All the input feature vectors in a corpus will form an input feature space, called the full feature space (FFS).

The FFS is then repeatedly presented to a SOM, whose topologic structure and learning algorithm are specified in Subsection 3.1, until the SOM reaches a steady convergence.

The SOM can be viewed as a two-layer artificial neural network, consisting of an input layer and an output layer. The output layer is a two-dimensional grid, each cell of which is associated with a weight vector whose dimension equals to the size of the feature space. There is a set of input data \( x_1, x_2, \ldots, x_n \) in the input layer, each input data is represented as a n-dimensional vector \( x_i = [x_{i1}, x_{i2}, \ldots, x_{in}]^T \). Each weight vector in the output layer is also represented as a n-dimensional vectors \( w_j = [w_{j1}, w_{j2}, \ldots, w_{jn}]^T \). The SOM defines a learning algorithm that maps the input vectors to the output layer according to the similarity between the \( x \) and \( w \). The learning process involves two main steps:

- **Selecting winning nodes**: The equation (4) defined by [9], called Tanimoto similarity is used to measure the similarity between input vectors and weight vectors. When input vector \( x \) is presented to the output layer, it is compared with all the \( w \) simultaneously by means of the parallel computing facilities. A node with the maximum similarity is defined the best-matching node (or winning node), denoted by \( c \) (see Equation (5)).

\[
s(x, w) = \frac{(x, w)}{\|x\|_2 + \|w\|_2 - (x, w)}
\]

where \( s(x, w) \) is the Tanimoto similarity between vectors \( x \) and \( w \); \( (x, w) \) is the dot product of \( x \) and \( w \); \( \|x\|_2 \) and \( \|w\|_2 \) are the Euclidean norms of \( x \) and \( w \) respectively.

\[
s(x, w) = m \min_{j} \left\{ s(x, w_j) \right\}
\]

- **Updating weight vectors**: The best-matching node \( c \) and its neighborly nodes, which are topographically close to the winning node in the grid up to a certain geometric distance, will learn something from the current input \( x \); their weights will be modified to make them more responsive to the current input via the following adjustments.

\[
w_{c(t+1)} = w_{c(t)} + \eta(t) \left[ s(x, w_c) - s(x, w_{c(t)}) \right]
\]

where \( t=0, 1, 2, \ldots \) is an integer, the discrete-time coordinate. The \( \eta(t) \) is the gain term. The value of \( \eta(t) \) is between 0 to 1. When \( t \rightarrow \infty, \eta(t) \rightarrow 0 \), the learning process is then converged.

The two major learning steps, selecting winning nodes and updating weight vectors, should be iterated many times, say, several thousands or even tens thousands times until a steady convergence is obtained.

When the SOM is converged the input data has been classified into clusters on its output layer, i.e. similar input vectors will be mapped onto adjacent areas of the output layer. The trained SOM can be used for retrieval.

### IV. EVALUATION

A prototype system, called Software Self-Organizing Map (SSOM) was established to test the effectiveness of the proposed input data representation scheme. A simulation on the prototype system was conducted. As a result of the simulation, a software self-organizing map (SSOM) was built. Totally 97 manual pages of the most useful UNIX system commands are chosen as the sample data in this simulation. The content identifiers, or features, associated with each manual page are extracted during the automatic indexing stage. A total of 827 features. A retrieval experiment based on the 827-dimensional FFS was conducted to evaluate the effectiveness of the proposed indexing method.

A total of 14 Queries, obtained from MS-DOS commands, were used in this experiment. Under MS-DOS system, typing a command followed by the ‘?’ switch displays a screen of explanation on that command in terms of its functionality. It is considered that these queries are more objective as these descriptions were not specifically generated for this retrieval experiment. Thus the retrieval results will be more convincing. These queries described by natural language (English) were transformed to query vectors by automatic indexing which is similar to the process of indexing software documents. Each query vector was presented to the trained SSOM and mapped to a winning node based on the Tanimoto similarity between
the query vector and the input feature vectors. Software components attached to the winning node and/or the nodes adjacent to the winning node were selected as candidates for the specified query.

When a SSOM is trained to classify a document collection it is used to retrieve the desired documents from the collection. Each retrieval query will be represented as a query vector that will be compared with all the weight vectors to determine which node in the SSOM is the best matched one. The documents located on the matched node and its adjacent nodes will be selected as retrieval candidates.

The retrieval experience based on a document collection of 97 UNIX manual pages showed very promising results, which supports the author’s view that the input data representation scheme for training the SSOM is effective and efficient. In future, experiments on larger document collections will be conducted.

V. CONCLUSIONS

An input data representation scheme for the SSOM is presented, which used an automatic indexing method. In the process of automatic indexing, phrases are identified in addition to the single descriptors, and a domain dependent relational thesaurus is built. Phrases in general carry more specific semantic meanings than the single words separately. This is an effective way to enhance retrieval precision. While terms in the thesaurus classes can replace the initial search terms, and thus can be used to increase retrieval recall.

The result obtained from the retrieval experiment is presented in Figure 1. Retrieval precisions achieved at different levels of recall are very promising. When the value of recall increases no significant decline of the value of the precision was observed.

![Figure 1. Retrieval result](image)

Figure 1. Figure. 1. Retrieval result

REFERENCES