CONCEPT BASED CLUSTERING FOR DESCRIPTIVE DOCUMENT CLASSIFICATION

S. Senthamarai Kannan[1*] ***and N. Ramaraj***[2]

[*1]*Department of Information Technology, Thiagarajar College of Engineering, Madurai, India*
Email: sskit@tce.edu
[2]*Principal, GKM Engineering College, Chennai, India*

ABSTRACT

We present an approach for improving the relevance of search results by clustering the search results obtained for a query string with the help of a Concept Clustering Algorithm. The Concept Clustering Algorithm combines common phrase discovery and latent semantic indexing techniques to separate search results into meaningful groups. It looks for meaningful phrases to use as cluster labels and then assigns documents to the labels to form groups. The labels assigned to each document cluster provide meaningful information on the various documents available under that cluster. This provides a more interactive and easier way to probe through search results and identifying the relevant documents for the users using the search engine.

Keywords: Web search, Concept-based clustering, Document classification, Vector space model, Singular value decomposition, Latent semantic indexing

1 INTRODUCTION

Search engines provide fast access to the information people request on the web. All popular search engines return links to Web pages matching the user's question rather than the question's answer. Luckily, this works most of the time because people tend to place questions together with answers. When the query sent to a search engine is efficient, the search engine picks up the right information that is requested by the user. The real problem arises when the information need expressed by the query is vague, too broad, or simply ill defined.

For an ill-defined or vague query, the range of covered topics becomes unmanageably large, confusing, and of doubtful value to the user. Search-results clustering (Osinski & Weiss, 2003) aims to present information about the matching documents. While clustering the search results, most of the work is done on identifying some underlying semantic relationship among the documents capable of explaining why these documents constitute a good result to the query rather than individual documents fetched by search engine.

The semantic structure or relationship among the documents plays a vital role in classifying the relevance of the document to the query. The documents that are more relevant to the query have the same semantic structure. To find this structure, we set the following procedures:

a) Identify groups of similar documents,
b) Discover a textual description of the property making the documents similar, and
c) Present these descriptions to the user in the document.

While presenting a document, the search result-clustering algorithm should speak the user's language to produce groups with descriptions easy for the users to read and understand. Users will likely disregard groups with overly long or ambiguous descriptions, even though their content might be valuable. A Web search-clustering algorithm must therefore aim to generate only clusters possessing meaningful, concise, and accurate labels.

Concept clustering follows a description-comes-first approach, in which careful selection of label candidates is crucial. The algorithm must ensure that labels are significantly different while covering most of the topics in the input snippets. To find such candidates, we use the vector space model (VSM) (Salton & McGill, 1983; Salton, et al., 1975) and singular value decomposition (SVD), the latter being the fundamental mathematical construct underlying the latent semantic indexing (LSI) technique (Littman, Dumais, & Landaur, 1996).

VSM is a method of information retrieval that uses linear-algebra operations to compare textual data. VSM associates a single multi-dimensional vector with each document in the collection, and each component of that vector reflects a particular key word or term related to the document. This helps in representing a set of documents by arranging their vectors in a term-document matrix. The value of a single component of the term-document matrix depends on the strength of the relationship between its associated term and the respective document.

Unlike VSM, LSI aims to represent the input collection using concepts found in the documents rather than the literal terms appearing in them. To do this, LSI approximates the original term-document matrix using a limited number of orthogonal factors. These factors represent a set of abstract concepts, each conveying some idea common to a subset of the input collection. From concept clustering's viewpoint, these concepts are perfect cluster label candidates; unfortunately, however, this matrix representation of LSI is difficult for people to understand. To obtain concise and informative labels, the verbal meaning behind the vector representation of the LSI-derived abstract concepts must be extracted.

Among all term co-occurrences in a collection of documents, phrases that appear at least a certain number of times in the input collection seem to be most meaningful to users and, at the same time, are relatively inexpensive to find. These phrases are often collocations or proper names, making them both informative and concise. Therefore these phrases can be used to approximate the verbal meaning of the SVD-derived abstract concepts. Concept clustering uses these frequent phrases as cluster labels. After discovering these diverse cluster labels, we assign documents to them using standard VSM as shown by the generalized algorithm in Figure 1.

```
/** Phase 1: Preprocessing */
For each document {
Do text filtering;
Identify the document's language;
Apply stemming;
Mark stop words;
}

/** Phase 2: Feature extraction */
Discover frequent terms and phrases;
/** Phase 3: Cluster label induction */
Use LSI to discover abstract concepts;
For each abstract concept
{Find best-matching phrase;}
Prune similar cluster labels;
/** Phase 4: Cluster content discovery */
For each cluster label
{Use VSM to determine the cluster contents;}
/** Phase 5: Final cluster formation */
Calculate cluster scores;
Apply cluster merging;
```

Figure 1. Pseudo code for various phases of the algorithm

2 VECTOR SPACE MODEL

The theoretical model for a text classifier (Bharati, Chaitanya, et al., 2002; Guthrie & Walker, 1994), similar to other pattern classification systems, is the vector space model. Each document is represented as a vector in a feature space. The features used are usually the words (terms) that make up the document. The component of a document vector along a feature dimension is called the weight of the term in that document. In document space, the categories are represented as clusters of their document vectors, usually as simple mean vectors of their document vectors. When a test document is given to the system, it computes different similarity measures between the test vector and each of the category vectors. The system then assigns the document to the most similar category. Some classifiers have more complex decision mechanisms. They use support vector machines, KNN classifiers, regression models, symbolic rule learning, etc. Most of them, however, have an underlying vector space model for the categories, and use a weighing formula for representing the category vectors. Several measures are used to assign weights, and each of them relies on a heuristic – inverse document frequency, maximizing information gain, etc.

It was observed by Salton that TfIdf vectors improve the space density measure more than the simple TF vectors, making the document vectors easier to classify. Further, the category vectors are represented as the mean vectors of their documents.

3 SINGULAR VALUE DECOMPOSITION AND LSI

The use of the singular value decomposition (SVD) has been proposed for text retrieval in several recent works (Osiniski, et al, 2003). This technique uses SVD to project a very high dimensional document and query vectors into a low dimensional space. In this new space, it is hoped that the underlying structure of the collection is revealed thus enhancing retrieval performance. In text retrieval, a simple way to represent a collection of documents is with a term-document matrix D, where D (i; j) gives the number of occurrences of term i in document j. Queries (over the same set of terms) are similarly represented. The similarity between document vectors (the columns of term-document matrices) can be found by their inner product. This corresponds to determining the number of term matches (weighted by frequency) in the respective documents. Another commonly used similarity measure is the cosine of the angle between the document vectors. This can be achieved computationally by first normalizing (to 1) the columns of the term-document matrices before computing inner products.

The frequency counts were used as the entries of the term-document matrix. In practice these counts are typically scaled using various term weightings Yamamoto & Masuyama, 1995), in order to cancel out the dominating effects of frequent terms. One scheme that is commonly used is inverse document frequency (IDF) weighting (Tokunaga & Iwayama, 1994). Latent semantic indexing (Kontostathis & Pottenger, 2002; Zha, 1998) attempts to project term and document vectors into a lower dimensional space spanned by the true factors of the collection. This uses a truncated singular value decomposition of the term-document matrix D. If D is an m X n matrix, then the SVD of D is D = USV^T where U is mXn with orthonormal columns, V is nXn with orthonormal columns, and S is diagonal with the main diagonal entries sorted in decreasing order.

4 ILLUSTRATION OF CLUSTER LABEL INDUCTION

a) List of Documents
D1: Large-scale singular value computations
D2: Software for the sparse singular value decomposition
D3: Introduction to modern information retrieval
D4: Linear algebra for intelligent information retrieval
D5: Matrix computations
D6: Singular value cryptogram analysis
D7: Automatic information organization
b) List of Keywords
T1: Information
T2: Singular
T3: Value
T4: Computations
T5: Retrieval
c) List of Phrases
P1: Singular value
P2: Information retrieval

Figure 2. Input data in the label induction phase

After considering the algorithms mentioned above, let us illustrate the ideas introduced, we analyze how concept clustering deals with an example collection of d = 7 document title (figure 2), in which t = 5 terms (figure 2) and p = 2 phrases (figure 2) appear more than once and thus are treated as frequent.

First, concept clustering builds a term-document matrix from the input snippets. In such a matrix, a column vector represents each snippet, and row vectors denote terms selected to represent document features. In our example, the first row represents the word “information,” and the second row represents “singular.” Similarly, column one represents D1 in figure 2, “large-scale singular value computations,” column two represents D2, “software for the sparse singular value decomposition,” and so on.

At this stage we disregard terms marked as stop words and terms that have not appeared more than a certain number of times. This not only significantly increases the algorithm’s time efficiency but also reduces noise among cluster labels. We use the popular Salton’s tfidf term-weighting scheme 4 to eliminate strong bias toward the query words in the snippets. We calculate values in matrix A using *tfidf* and then normalize each column vector’s length. We base the actual process of discovering abstract concepts on the SVD of the term document matrix A, which decomposes into three matrices (*U*, *S*, and *V*), such that $A = USV^T$.

One of the SVD’s properties is that the first *r* columns of U (where *r* is the rank of A) form an orthogonal basis for the input matrix’s term space. In linear algebra, the basis vectors of a linear space can serve as building blocks for creating vectors over that space. Using this knowledge, in our setting, each building block should carry one of the ideas referred to in the input collection. Thus, from concept clustering’s viewpoint, basis vectors (that is, the column vectors in *U*) are exactly what it has set out to find - a vector representation of the snippets’ abstract concepts. In most practical situations, taking all *r* basis vectors as abstract concepts would result in an unmanageable number of clusters - usually close to the number of snippets. Therefore, concept clustering uses the singular values of the *A* matrix (lying on the diagonal of the SVD’s *S* matrix) to calculate how many columns of *U* should actually proceed to the next stage of the algorithm.

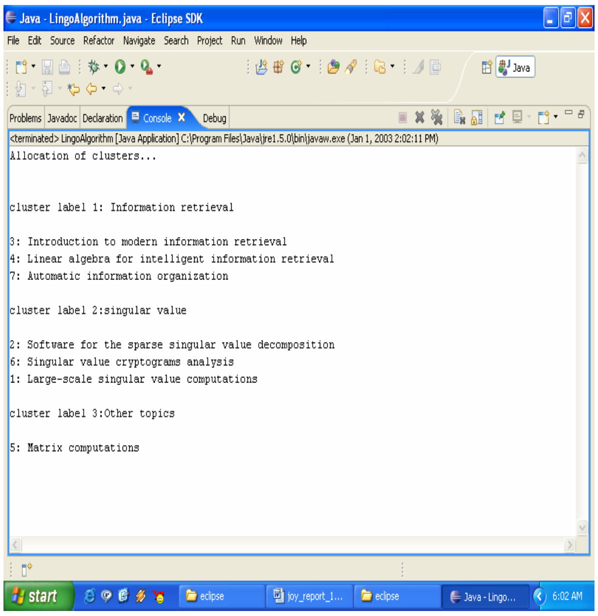
Assume that the calculation results in k = 2 are set as the desired number of clusters for our example. Consequently, in further processing, we use U_k, which consists of the first *k* columns of *U*. Two characteristics of the basis vectors are important here. First, the vectors are pair-wise orthogonal, which should result in a rich diversity among the discovered abstract concepts. Second, basis vectors are expressed in the A matrix's term space and thus can be the frequent phrases discovered in the previous phase.

Concept clustering can therefore use the classic cosine distance measure to compare and calculate which phrase or word will be the best verbal representation of an abstract concept. We consider single terms at this stage because it is possible that none of the frequent phrases describes an abstract concept better than a single term. For every frequent phrase and single frequent word, concept clustering creates a column vector over the A matrix's term space. When assembled together, these vectors form a term-document matrix P of frequent phrases and words. Assuming column vectors of both U and P are length-normalized, the problem of calculating the cosine distance between every abstract-concept–phrase-or-term pair reduces to simple matrix multiplication. Rows of the resulting matrix M represent abstract concepts, its columns represent phrases and single words, and individual values are the cosine similarities in question. Thus, in a single row, the maximum component indicates the phrase or single word that best approximates the corresponding abstract concept. As the values in the M matrix range from 0.0 (no relationship) to 1.0 (perfect match), concept clustering also uses the maximum components as cluster label scores.

5 CLUSTER CONTENT ALLOCATION

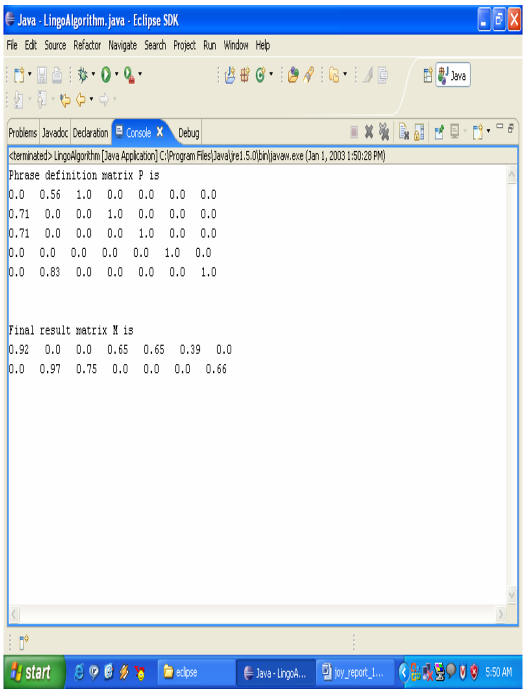
The cluster-content allocation process resembles VSM-based document retrieval except that instead of one query, Concept clustering matches the input snippets against a series of queries, each of which is a single cluster label. So, for a certain query label, if the similarity between a snippet and the label exceeds a predefined threshold (snippet assignment threshold), Concept clustering allocates the snippet to the corresponding cluster. Snippet assignment threshold values fall within the 0.0–1.0 range. Higher threshold values result in more documents being put in clusters, which can decrease the assignment precision. Lowering the value leads to lower recall. The snippet assignment threshold value is therefore largely a matter of user preference. We have empirically verified that thresholds within the 0.15–0.30 range produce the best results. This assignment scheme naturally creates overlapping clusters and nicely handles cross-topic documents. We can also use the similarity values to sort snippets within their groups, making the most relevant snippets easier to identify. Finally, we created an "other topics" group for those snippets that do not match any of the cluster labels. The last operation of the cluster-content allocation phase is calculating group scores as a product of the label score and the number of snippets in the group.

6 EXPERIMENTAL RESULTS AND DISCUSSION


Java - LingoAlgorithm.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Problems Javadoc Declaration Console Debug
start eclipse joy_report_1... eclipse Java - Lingo... 6:02 AM

```
<terminated> LingoAlgorithm [Java Application] C:\Program Files\Java\jre1.5.0\bin\javaw.exe (Jan 1, 2003 2:02:11 PM)
Allocation of clusters...


cluster label 1: Information retrieval

3: Introduction to modern information retrieval
4: Linear algebra for intelligent information retrieval
7: Automatic information organization

cluster label 2:singular value

2: Software for the sparse singular value decomposition
6: Singular value cryptograms analysis
1: Large-scale singular value computations

cluster label 3:Other topics

5: Matrix computations
```

Figure 3a. Construction of the phrase definition matrix and final result matrix


Java - LingoAlgorithm.java - Eclipse SDK
File Edit Source Refactor Navigate Search Project Run Window Help
Java
Problems Javadoc Declaration Console Debug
start
eclipse
Java - LingoA...
joy_report_1...
5:50 AM

```
<terminated> LingoAlgorithm [Java Application] C:\Program Files\Java\jre1.5.0\bin\javaw.exe (Jan 1, 2003 1:50:28 PM)
Phrase definition matrix P is
0.0   0.56   1.0   0.0   0.0   0.0   0.0
0.71   0.0   0.0   1.0   0.0   0.0   0.0
0.71   0.0   0.0   0.0   1.0   0.0   0.0
0.0   0.0   0.0   0.0   0.0   1.0   0.0
0.0   0.83   0.0   0.0   0.0   0.0   1.0


Final result matrix M is
0.92   0.0   0.0   0.65   0.65   0.39   0.0
0.0   0.97   0.75   0.0   0.0   0.0   0.66
```

Figure 3b. Cluster contents for browsing

In the cluster label induction phase, a significant amount of matrix computation (mostly transpositions and multiplications) is performed. We have decided to use the external library JAMA (http:math.nist.gov/javanumerics/jama/doc) to avoid an independent implementation of single value decomposition. Experiments were done on an Athlon1.3GHz/768MB machine that found the single value decomposition accounts for over 70% of the total computational expense. Noticeable is also the insignificant contribution of the feature discovery phase towards the overall clustering time.

7 FUTURE IMPROVEMENTS

We discuss the future improvements of concept clustering, which are possible remedies for the above limitations. While practical experiments suggest that it performs fairly efficiently in Web search results clustering tasks, several extensions could be introduced to ensure even better quality of grouping. Below we give an overview of the proposed improvements.

A. Hierarchical clustering

Owing to its flexible design, there are several ways in which the algorithm could be extended to support hierarchical groupings. A simple method based on cluster content overlap could be added as an additional step to concept clustering's post-processing phase. Another approach could utilize the fact that, apart from document-query similarities, latent semantic indexing can calculate a matrix of term-term (or phrase-phrase) associations. Based on such a matrix, not only could the hierarchical structure rely on the actual content of groups but also on the semantic relationships between their labels.

B. Incremental processing

In order to achieve a noticeable improvement in concept clustering's response time, an incremental model of processing would have to be introduced primarily at the singular value decomposition stage. This, however, requires that all preceding phases must also perform their computations in a similar manner.

The preprocessing phase is by its very nature a sequential process, and hence it should not present any difficulties while designing an incremental version. The feature extraction phase, however, is more difficult to modify. In the present implementation of concept clustering, phrase discovery is based on suffix arrays. Therefore an incremental method of building the data structure would be needed. In suffix array construction, techniques split large input texts into smaller blocks. In this way, suffix sorting can be performed on each block separately, but at the same time, partial results can be successively merged into the final suffix array. An incremental implementation of the cluster label induction phase requires that the latent semantic indexing be performed in the same way.

Two techniques – Folding-In and SVD-Updating – are presented that allow adding new documents to an already built k-rank approximation of the term document matrix without recomputing the SVD from scratch. While the accuracy of these methods is slightly worse than the explicit SVD recomputation, they are much more time-efficient. As it is the singular value decomposition that accounts for more than 70% of concept clustering's running time, an alternative probabilistic SVD algorithm could eliminate the need for incremental label induction. Another approach could employ concept indexing – an effective dimensionality reduction method based on the k-means algorithm.

8 CONCLUSION

Web search results clustering are gaining increasingly promising perspectives for the future. On the one hand, extensive research is being conducted on faster and more precise clustering algorithms, and on the other, scientific advances are followed by practical implementations of the idea, some of which, such as Vivisimo, have already achieved commercial success. We aimed to contribute to both the scientific and the practical trend in web search clustering. Analyzing the currently available algorithms, we observed that little emphasis was being placed on the quality of thematic groups description. This was the major incentive for us to propose and implement a new method whose main priority is to provide concise and meaningful group labels.

9 REFERENCES

Bharati, A., Chaitanya, K., et al. (002) *Document Space Model for Automated Text Classification based on Frequency Distribution across Categorie*s. International Institute of Information Technology technical report.

Guthrie, L. & Walker, E. (1994) Document Classificationby Machine: Theory and Practice. *Proceedings of COLING 94,* 1059-1063.

Java website. Retrieved from the WWW, August 20, 2007: http://math.nist.gov/javanumerics/jama/doc

Kontostathis, A. & Pottenger, W. (2002) A Mathematical View of Latent semantic indexing: Tracing Term Co-ocurrences.

Littman, M., Dumais, S., & Landauer, T. (1996) Automatic Cross-Language Information Retrieval using Latent semantic indexing. *Proceedings of SIGIR-96*, Zurich.

Osinski, S., Stefanowski, J., & Weiss, D. (2003) Concept clustering: Search results clustering algorithm based on Singular Value Decomposition. *Intelligent Information Systems Conference 2004*, Zakopane, Poland.

Osinski, S. & Weiss, D. (2004) Lingo: *A Search Results Clustering Algorithm based on Singular Value Decomposition*. Poznan University of Technology.

Salton, G. & McGill, M. (1983) *Introduction to Modern Information Retrieval*. McGraw-Hill.

Satlon, G., Wong, A., & Yang, C. (1975) Cornell University A Vector Space Model for Automatic Indexing. *Association of Computing Machines.*

Thorsten, J. (1998) Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Proceedings of ECML-98*

Tokunaga, T. & Iwayama, M. (1994) Text *categorization based on weighted inverse document frequency*. Department of Computer Science, Tokyo Institute of Technology technical report.

Yamamoto, K. & Masuyama, S. (1995) Automatic Text Classification Method with Simple Class- Weighting Approach. *Natural Language Processing Pacific Rim Symposium*.

Yang, Y. & Pedersen, J. (1997) A Comparative Study on Feature Selection in Text Categorization. *Proceedings of ICML-97*

Zamir, O. (1999) *Clustering Web Documents: A Phrase-Based Method for Grouping Search Engine Results*. Doctoral Dissertation, University of Washington, Seattle, WA.

Zha, H. (1998) A subspace-based model for information retrieval with applications in Latent semantic indexing. Proceedings of Irregular '98, Lecture *Notes in Computer Science 1457*, pp. 29-42 Springer-Verlag.