ISSN No. 0976-5697

Volume 3, No. 2, March-April 2012

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Web Snippet Clustering and Labeling using Lingo Algorithm

B.R.Prakash*
Research Scholar, Department of Computer Science & Applications, Bangalore University, Bangalore,India
brp.tmk@gmail.com

Hanumanthappa.M
Professor, Department of Computer Science & Applications, Bangalore University, Bangalore, India
hanu6572@hotmail.com

Abstract: A typical search engine's response to a query is a ranked list of documents along with their partial content (snippets) Search results clustering problem is defined as an automatic, on-line grouping of similar documents in a search hits list, returned from a search engine. In this paper we discuss how to cluster the snippets and generate the label for them, further the results of the experimental evaluation of a Lingo algorithm using singular value decomposition.

Keywords: Clustering; Snippets; Cluster Label; Candidates words; Pre-processing

I. INTRODUCTION

Clustering is an important operation in the exploratory analysis of large data sets. Given a data set from some domain, it is sometimes desirable to group data items that are similar to each other under the same cluster and data items that are dissimilar from each other under different clusters. As a consequence, the subdivision of the data set into clusters makes the latent structure of the data space explicit, and facilitates further human or automatic analysis. Tools for clustering the results returned by Web search engines have recently become a focus of attention in the IR research community. Real-time clustering technology is a key ingredient of such systems, since the partition of the search results into clusters must be generated on-the-fly. Clustering search results differs significantly from other types of document clustering. Each matching result (hit) in a list of results returned by a search engine contains a resource locator (URL), an optional title, and a short fragment of text called a snippet [1][6], Two or three best matching spans are joined and returned as a short block of text providing insight into the original document for the user. This technique of generating snippets is called kwic — *keyword in context*. Figure 1 shows a typical snippet.

Carrot2 - Wikipedia, the free encyclopedia
en.wikipedia.org/wiki/**Carrot2**
Carrot² is an open source search results clustering engine. It can automatically cluster small collections of documents, e.g. search results or document abstracts, ...

Figure 1: A typical "hit" returned by a search engine: document title on top, snippet with query terms in the middle and an information line with the document's address

II. OVERVIEW OF DESCRIPTION COMES FIRST APPROACH

The DCF changes the troublesome conventional order of a typical clustering algorithm (cluster discovery → label induction) in a way that splits cluster discovery and candidate label discovery into two *independent* phases

a. Candidate label discovery is responsible for collecting all phrases potentially useful as good cluster labels.
b. Cluster discovery provides a computational data model of document groups present in the input data.

By splitting the process into these two phases the most difficult element so far — creating proper cluster description from a mathematical model—is avoided and replaced by a problem of *selection* of appropriate labels for each cluster found.

The DCF approach tries to decrease this "semantic gap" between a model of clusters and a set of selected labels by discarding the model of clusters entirely and building final groups of documents starting with just the selected cluster labels. This way the actual model of clusters is used only internally and can be suitably complex because it never surfaces to the user interface. On the other hand the candidate cluster label selection procedure should ensure their comprehensibility to the user.

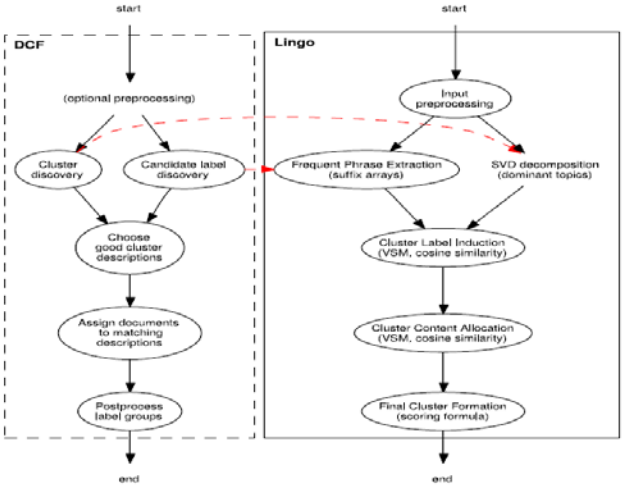

Figure 2: Generic elements of DCF and their counterparts in Lingo. svd decomposition takes place inside cluster label induction phase, it is extracted here for clarity.

III. OVERVIEW OF THE ALGORITHM

Lingo processes the input in four phases: snippets preprocessing, frequent phrase extraction, cluster label induction and content allocation.

A. Input Preprocessing:

In the preprocessing phase the input documents (titles and snippets) are tokenized and split into terms. Lingo is implemented as a component embedded in the Carrot2 framework. and uses its infrastructure to perform certain text preprocessing tasks — stemming, marking stop words and simple text segmentation heuristics. After tokenization is complete, a term-document matrix is constructed out of the terms that exceed a predefined term frequency threshold. After that, document vectors are weighted using the tf-idf formula [2].Terms present in document titles are additionally boosted compared to these appearing in snippets by a predefined constant because titles are more likely to contain sensible (human edited) information.

a. *D* ←input documents (or snippets)
   /* **Preprocessing** */
b. **for all** *d* ∈*D* **do**
c. perform text segmentation of *d*; /* Segmentation, stemming. */
d. **if** language of *d* recognized **then**
e. apply stemming and mark stop-words in *d*;
f. **end if**
g. **end for**
   /* **Frequent Phrase Extraction** */
h. concatenate all documents;
i. P*c* ← discover complete phrases;
j. P*f* ←*p* : {*p* ∈P*c* ∧frequency(*p*) > *term frequency threshold*};
   /* **Cluster Label Induction** */
k. *A*← term-document matrix of terms not marked as stop-words and with frequency higher than the *Term Frequency Threshold*;
l. *S*,*U*,*V* ←SVD(*A*); /* Product of SVD decomposition of *A* */
m. *k* ←0; /* Start with zero clusters */
n. *n*←*rank*(*A*);
o. **repeat**
p. *k* ←*k* +1;
q. *q* ←(°°*Sk*°°*F* /k*S*k*F*);
r. **until** *q* < *Candidate Label Threshold*;
s. *P* ← phrase matrix for P*f* ;
t. **for all** columns of *UT*
   k
   P **do**
u. find the largest component *mi* in the column;
v. add the corresponding phrase to the *Cluster Label Candidates* set;
w. *labelScore*←*mi* ;
x. **end for**
y. calculate cosine similarities between all pairs of candidate labels;
z. identify groups of labels that exceed the *Label Similarity Threshold*;
aa. **for all** groups of similar labels **do**
bb. select one label with the highest score; /* cluster description */
cc. **end for**
   /* **Cluster Content Discovery** */
dd. **for all** *L* ∈*Cluster Label Candidates* **do**
ee. create cluster *C* described with *L*;
ff. add to *C* all documents whose similarity to *C* exceeds the *Snippet Assignment Theshold*;
gg. **end for**
hh. put all unassigned documents in the "Others" group;
   /* **Final Cluster Formation** */
ii. **for all** clusters **do**
jj. *clusterScore*←*labelScore*×k*C*k;
kk. **end for**
ll. Sort final clusters.

Algorithm 1: Pseudo-code of the Lingo algorithm.

B. Frequent Phrase Extraction:

The aim of this step is to discover a set of cluster label candidates — phrases (but also single terms) that can potentially become cluster labels later. Lingo extracts frequent phrases using a modification of an algorithm presented in the SHOC algorithm [3]. A word-based suffix array is constructed and extended with an auxiliary data structure — the LCP (Longest Common Prefix). This allows the algorithm to identify all frequent complete phrases in O(n) time, n being the total length of all input snippets. The frequent phrase extraction algorithm ensures that the discovered labels fulfill the following conditions:

a. appear in the input at least a given number of times (it is a tuning threshold);
b. not cross sentence boundaries; sentence markers indicate a topical shift, therefore a phrase extending beyond one sentence is unlikely to be meaningful;
c. be a complete frequent phrase (the longest possible phrase that is still frequent); compared to partial phrases, complete phrases should allow clearer description of clusters
d. neither begin nor end with a stop word; stop words that appear in the middle of a phrase should not be discarded.

C. Cluster Label Induction:

During the cluster label induction phase, Lingo identifies the abstract concepts (or dominant topics in the terminology used in DCF) that best describe the input collection of snippets. There are two steps to this: abstract concept discovery, phrase matching and label pruning.

In abstract concept discovery, singular value decomposition (SVD) is applied to the term document matrix A, breaking it into three matrices: U, S and V in such a way that A = USV T . An interesting property of SVD is that the first r columns of matrix U, r being the rank of A, form an orthogonal basis for the term space of the input matrix A [4]. It is commonly believed that base vectors of the decomposed term-document matrix represent an approximation of "topics" —collections of terms connected with an obscure net of latent relationships.

Although this fact is difficult to prove, singular decomposition is widely used in text processing, for example in Latent Semantic Indexing (LSI). From Lingo's point of view, basis vectors (column vectors of matrix U) contain exactly what it has set out to find — a vector representation of the abstract concepts.

The most significant k base vectors of matrix U are determined by selecting the Frobenius norms (measuring the difference between two matrices) of the term-document matrix A and its k-rank approximation AK . Let threshold q be a percentage-expressed value that determines to what extent the k-rank approximation should retain the original information in matrix A. We hence define k as the minimum value that satisfies the following condition:

$$||AK||F/||A||F \geq q$$

Where the symbol ||X||F denotes the Frobenius norm of matrix X. Clearly, the larger the value of q the more cluster candidates will be induced. The choice of the optimal value for this parameter ultimately depends on the preferences of users, so we make it one of Lingo's control thresholds — Candidate Label Threshold.

Phrase matching and label pruning step, where group descriptions are discovered, relies on an important observation that both abstract concepts and frequent phrases are expressed in the same vector space — the column space of the original term-document matrix A. This enables us to use the cosine distance to calculate how "close" a phrase or a single term is to an abstract concept. Let us denote by P a matrix of size $t \times (p + t)$, where t is the number of frequent terms and p is the number of frequent phrases. P can be easily built by treating phrases as pseudo-documents and using one of the term weighting schemes. Having the P matrix and the i -th column vector of the SVD's U matrix, a vector mi of cosines of the angles between the i -th abstract concept vector and the phrase vectors can be calculated as:

$$mi = Ui^T P.$$

The phrase that corresponds to the maximum component of the mi vector should be selected as the human-readable description of i -th abstract concept. Additionally, the value of the cosine (similarity) becomes the score of the cluster label candidate. A similar process for a single abstract concept can be extended to the entire Uk matrix —a single matrix multiplication M =UkT P yields the result for all pairs of abstract concepts and frequent phrases. The final step of label induction is to prune overlapping labels. Let V be a vector of cluster label candidates and their scores. We create another term-document matrix Z, where cluster label candidates serve as documents. After column length normalization we calculate ZT Z, which yields a matrix of similarities between cluster labels. For each row we then pick columns that exceed the Label Similarity Threshold and discard all but one cluster label candidate with the maximum score which becomes the description of a future cluster.

IV. CLUSTER CONTENT ALLOCATIONS

The process of cluster content allocation very much resembles document retrieval based on plain VSM model. The only difference is that instead of one query, the input snippets are matched against a series of queries, each of which is a single cluster label. Thus, if for a certain query-label, the similarity between a document and the label exceeds a predefined threshold, it will be allocated to the corresponding cluster. Note that from the point of view of DCF traditional Vector Space Model used for comparisons is not ideal — the label's word order and proximity is not taken into account. Let us define matrix Q, in which each cluster label is represented as a column vector. Let C =QT A, where A is the original term-document matrix for input documents. This way, element Ci j of the C matrix indicates the strength of membership of the j th document to the i -th cluster. A document is added to a cluster if Ci j exceeds the Snippet Assignment Threshold, yet another control parameter of the algorithm. Documents not assigned to any cluster end up in an artificial cluster called "Other documents".

V. FINAL CLUSTER FORMATION

Finally, clusters are sorted for display based on their score, calculated using the following formula:

$$C_{score} = \text{label score} \times ||C||,$$

Where ||C|| is the number of documents assigned to cluster C. The scoring function, although simple, prefers well-described and relatively large groups over smaller ones.

A. An Illustrative Example:

Let the input collection of documents contain d = 7 documents. We omit the preprocessing stage and assume t = 5 terms and p = 2 phrases are given (these appear more than once and thus will be treated as frequent). The input is shown in below.

The t = 5 terms
T1: Information
T2: Singular
T3: Value
T4: Computations
T5: Retrieval
The p = 2 phrases
P1: Singular Value
P2: Information Retrieval
The d = 7 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 Analysis of Cryptograms
D7: Automatic Information Organization

Figure 5.3: Input documents, frequent terms and phrases

We now preprocess the input term document matrix — tf-idf weighting and normalization results in matrix Atf-idf, SVD decomposition of that matrix yields matrix U containing abstract concepts.

$$A_{tfidf} = \begin{bmatrix} 0 & 0 & 0.56 & 0.56 & 0 & 0 & 1 \\ 0.49 & 0.71 & 0 & 0 & 0 & 0.71 & 0 \\ 0.49 & 0.71 & 0 & 0 & 0 & 0.71 & 0 \\ 0.72 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.83 & 0.83 & 0 & 0 & 0 \end{bmatrix}$$

$$U = \begin{bmatrix} 0 & 0.75 & 0 & -0.66 & 0 \\ 0.65 & 0 & -0.28 & 0 & -0.71 \\ 0.65 & 0 & -0.28 & 0 & 0.71 \\ 0.39 & 0 & 0.92 & 0 & 0 \\ 0 & 0.66 & 0 & 0.75 & 0 \end{bmatrix}$$

Now we look for the value of k — the estimated number of clusters. Let us define quality threshold q = 0.9. Then the process of estimating k is as follows:

$$k = 0 \rightarrow q = 0.62,\ k = 1 \rightarrow q = 0.856,$$
$$k = 2 \rightarrow q = 0.959$$

And the number of expected clusters is k = 2.

To find relevant descriptions of our clusters (k = 2 columns of matrix U), we calculate similarity between candidate phrases and concept vectors as matrix M =Uk T P, where P is a synthetic term-document matrix created out of our frequent phrases and terms (values in matrix P are again weighted using tf-idf and normalized):

$$P = \begin{bmatrix} 0 & 0.56 & 1 & 0 & 0 & 0 & 0 \\ 0.71 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0.71 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.83 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

$$M = \begin{bmatrix} 0.92 & 0 & 0 & 0.65 & 0.65 & 0.39 & 0 \\ 0 & 0.97 & 0.75 & 0 & 0 & 0 & 0.66 \end{bmatrix}$$

Rows of matrix M represent clusters, columns — their descriptions. For each row we select the column with maximum value. The two selected labels are: Singular Value (score: 0.92) and Information Retrieval (score: 0.97). We skip label pruning as it is not necessary in this example. Finally, documents are allocated to clusters by applying matrix Q, created out of cluster labels, back to the original matrix bAtf-idf. The final result is shown below. Note the fifth column in matrix C, representing unassigned document D5.

$$Q = \begin{bmatrix} 0 & 0.56 \\ 0.71 & 0 \\ 0 & 0 \\ 0 & 0.83 \end{bmatrix}$$

$$C = \begin{bmatrix} 0.69 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0.56 \end{bmatrix}$$

Information Retrieval [score: 1.0]
D3: Introduction to Modern *Information Retrieval*
D4: Linear Algebra for Intelligent *Information Retrieval*
D7: Automatic *Information* Organization
Singular Value [score: 0.95]
D2: Software for the Sparse *Singular Value* Decomposition
D6: *Singular Value* Analysis of Cryptograms
D1: Large Scale *Singular Value Computations*
Other: [unassigned]
D5: Matrix *Computations*

VI. CONCLUSION

The motivation for creating Lingo was to come up with an algorithm for clustering search results capable to discover diverse groups of documents and at the same time keep cluster labels sensible. The work on Lingo must be credited to Stanisław Osi ´nski who worked on the algorithm under supervision of Jerzy Stefanowski [5] and later contributed a great deal of effort to the Carrot2 framework. The aim of this paper is to show how Lingo fits in the general scheme introduced by the DCF. The algorithm is an example of DCF's application to the domain of search results clustering and several elements of its implementation are designed specifically to deal with this type of input data.

VII. REFERENCES

[1]. Stanisław Osi ´nski, Jerzy Stefanowski, and Dawid Weiss. Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition. In Proceedings of the International Intelligent Information Processing and Web Mining Conference, Zakopane, Poland, Advances in Soft Computing, pages 359–368. Springer, 2004.

[2]. Gerard Salton. Automatic Text Processing — The Transformation, Analysis, and Retrieval of Information by Computer. Addison-Wesley, 1989.

[3]. Zhang Dong. Towards Web Information Clustering. PhD thesis, Southeast University, Nanjing, China, 2002.

[4]. Gene H. Golub and Charles F. Van Loan. Matrix Computations. The Johns Hopkins University Press, London, third edition, 1996.

[5]. Stanisław Osi ´nski and Dawid Weiss. A Concept-Driven Algorithm for Clustering Search Results. IEEE Intelligent Systems, 20(3):48–54, 2005.

[6]. W. Aisha Banu, Dr. P. Sheikh Abdul KaderA "Hybrid Context based Approach for Web Information" Retrieval International Journal of Computer Applications (0975 – 8887) Volume 10– No.7, November 2010.