Lingo: Search Results Clustering Algorithm Based on Singular Value Decomposition

Stanisław Osiński, Jerzy Stefanowski, and Dawid Weiss

Institute of Computing Science, Poznań University of Technology,
ul. Piotrowo 3A, 60–965 Poznań, Poland,
E-mail: stanislaw.osinski@man.poznan.pl,
{jerzy.stefanowski, dawid.weiss}@cs.put.poznan.pl

Abstract. Search results clustering problem is defined as an automatic, on-line grouping of similar documents in a search results list returned from a search engine. In this paper we present Lingo—a novel algorithm for clustering search results, which emphasizes cluster description quality. We describe methods used in the algorithm: algebraic transformations of the term-document matrix and frequent phrase extraction using suffix arrays. Finally, we discuss results acquired from an empirical evaluation of the algorithm.

Knowledge is of two kinds: we know a subject ourselves,
or we know where we can find information about it.
— Samuel Johnson, 1775

1 Introduction and related work

With an enormous growth of the Internet it has become very difficult for the users to find relevant documents. In response to the user's query, currently available search engines return a ranked list of documents along with their partial content (*snippets*). If the query is general, it is extremely difficult to identify the specific document which the user is interested in. The users are forced to sift through a long list of off-topic documents. Moreover, *internal relationships* among the documents in the search result are rarely presented and are left for the user. One of the alternative approaches is to automatically group search results into thematic groups (*clusters*). *Clustering of web search results* was first introduced in the Scatter-Gather [4] system. Several algorithms followed; Suffix Tree Clustering, (STC), implemented in the Grouper system [11] pioneered in using recurring phrases as the basis for deriving conclusions about similarity of documents. MSEEC [3] and SHOC [2] also made explicit use of words proximity in the input documents. Apart from phrases, graph-partitioning methods have been used in clustering search results [6]. Vivisimo is an example of a successful commercial application of the clustering idea.

Readable and unambiguous descriptions of the thematic groups are an important factor of the overall quality of clustering. They provide the users

an overview of topics covered in the search results and help them to identify the specific group of documents they were looking for. We feel this problem has not been sufficiently solved in the previous research resulting in too long, ambiguous and very often even meaningless group labels.

In this paper we briefly present our novel algorithm Lingo, which we believe is able to capture thematic threads in a search result, that is discover groups of related documents and describe the subject of these groups in a way meaningful to a human. Lingo combines several existing methods to put special emphasis on meaningful cluster descriptions, in addition to discovering similarities among documents.

2 Theoretical background

Vector Space Model Vector Space Model (VSM) is a technique of information retrieval that transforms the problem of comparing textual data into a problem of comparing algebraic vectors in a multidimensional space. Once the transformation is done, linear algebra operations are used to calculate similarities among the original documents. Every unique term (word) from the collection of analyzed documents forms a separate dimension in the VSM and each document is represented by a vector spanning all these dimensions. For example, if vector v represents document j in a k-dimensional space Ω, then component t of vector v, where $t \in 1 \dots k$, represents the degree of the relationship between document j and a term corresponding to dimension t in Ω. This relationship is best expressed as a $t \times d$ matrix A, usually named a *term-document matrix*, where t is the number of unique terms and d is the number of documents. Element a_{ij} of matrix A is therefore a numerical representation of relationship between term i and document j. There are many methods for calculating a_{ij}, commonly referred to as *term weighting methods*. Refer to [9] for an overview. Once matrix A has been constructed, the distance between vectors representing documents a and b, can be calculated in a variety of ways; the most common measure calculates a cosine between a and b using vector dot product formula.

Suffix arrays Let $A = a_1 a_2 a_3 \dots a_n$ be a sequence of objects. Let us denote by A_i a suffix of A starting at position $i \in 1 \dots n$, such as $A_i = a_i a_{i+1} a_{i+2} \dots a_n$. An *empty suffix* is also defined for every A as $A_{n+1} = \emptyset$. A *suffix array* is an ordered array of all suffixes of A. Suffix arrays, introduced in [5], are used as an efficient data structure for verifying whether a sequence of objects B is a substring of A, or more formally: $\exists i : B = A_i$ (sequence equality is equality of elements at their corresponding positions in A and B). The complexity of this operation is $O(P + logN)$, a suffix array can be built in $O(NlogN)$.

Latent Semantic Indexing and Singular Value Decomposition LSI is a technique of feature extraction which attempts to reduce the rank of a term-frequency matrix in order to get rid of noisy or synonymous words and

exploit the underlying latent structure of concepts in documents [1]. An algebraic method of matrix decomposition called Singular Value Decomposition is used for discovering the orthogonal basis of the original term-document matrix. This basis consists of orthogonal vectors that, at least hypothetically, correspond to topics present in the original term-document matrix.

SVD breaks a $t \times d$ matrix A into three matrices U, Σ and V, such that $A = U\Sigma V^T$. U is a $t \times t$ orthogonal matrix whose column vectors are called the left singular vectors of A, V is a $d \times d$ orthogonal matrix whose column vectors are called the right singular vectors of A, and Σ is a $t \times d$ diagonal matrix having the singular values of A ordered decreasingly along its diagonal. The rank r_A of matrix A is equal to the number of its non-zero singular values. The first r_A columns of U form an orthogonal basis for the column space of A—an essential fact used by Lingo.

3 Overview of the Lingo algorithm

When designing a web search clustering algorithm, special attention must be paid to ensuring that both content and description (labels) of the resulting groups are meaningful to humans. As stated on Web pages of Vivisimo (`http://www.vivisimo.com`) search engine, "a good cluster—or document grouping—is one, which possesses a good, readable description". The majority of open text clustering algorithms follows a scheme where cluster content discovery is performed first, and then, based on the content, the labels are determined. But very often intricate measures of similarity among documents do not correspond well with plain human understanding of what a cluster's "glue" element has been. To avoid such problems Lingo reverses this process—we first attempt to ensure that we *can* create a human-perceivable cluster label and only then assign documents to it. Specifically, we extract frequent phrases from the input documents, hoping they are the most informative source of human-readable topic descriptions. Next, by performing reduction of the original term-document matrix using SVD, we try to discover any existing latent structure of diverse topics in the search result. Finally, we match group descriptions with the extracted topics and assign relevant documents to them.

Algorithm 1 presents Lingo in the form of pseudo-code. Specific steps of the algorithm are explained later in this section.

3.1 Preprocessing

Stemming and stop words removal are very common operations in Information Retrieval. Interestingly, their influence on results is not always positive—in certain applications stemming yielded no improvement to overall quality. Be as it may, our previous work [10] and current experiments show that preprocessing is of great importance in Lingo because the input snippets are

Algorithm 1 Pseudo-code of the Lingo algorithm

$D \leftarrow$ input documents (or snippets)

{**STEP 1:** Preprocessing}
for all $d \in D$ **do**
  perform text segmentation of d; {Detect word boundaries etc.}
  if language of d recognized **then**
    apply stemming and mark stop-words in d;
  end if
end for

{**STEP 2:** Frequent Phrase Extraction}
concatenate all documents;
$\mathcal{P}_c \leftarrow$ discover complete phrases; {See Section 3.2 for details}
$\mathcal{P}_f \leftarrow p : \{p \in \mathcal{P}_c \wedge \text{frequency}(p) > \textit{Term Frequency Threshold}\}$;

{**STEP 3:** Cluster Label Induction}
$A \leftarrow$ term-document matrix of terms not marked as stop-words and with frequency higher than the *Term Frequency Threshold*;
$\Sigma, U, V \leftarrow \text{SVD}(A)$; {Product of SVD decomposition of A}
$k \leftarrow 0$; {Start with zero clusters}
$n \leftarrow rank(A)$;
repeat
  $k \leftarrow k + 1$;
  $q \leftarrow (\sum_{i=1}^{k} \Sigma_{ii})/(\sum_{i=1}^{n} \Sigma_{ii})$;
until $q <$ *Candidate Label Threshold*;

$P \leftarrow$ phrase matrix for $\mathcal{P}_f$; {See section 3.3}
for all columns of $U_k^T P$ **do**
  find the largest component m_i in the column;
  add the corresponding phrase to the *Cluster Label Candidates* set;
  $labelScore \leftarrow m_i$;
end for

calculate cosine similarities between all pairs of candidate labels;
identify groups of labels that exceed the *Label Similarity Threshold*;
for all groups of similar labels **do**
  select one label with the highest score;
end for

{**STEP 4:** Cluster Content Discovery}
for all $L \in$ *Cluster Label Candidates* **do**
  create cluster C described with L;
  add to C all documents whose similarity to C exceeds the *Snippet Assignment Theshold*;
end for
put all unassigned documents in the "Others" group;

{**STEP 5:** Final Cluster Formation}
for all clusters **do**
  $clusterScore \leftarrow labelScore \times \|C\|$;
end for

automatically generated summaries of the original documents and hence are usually very small (one or two sentences). Although SVD is capable of dealing with noisy data, without sufficient preprocessing, the majority of discovered abstract concepts would be related to meaningless frequent terms. The aim of the preprocessing phase is to prune from the input all characters and terms that can possibly affect the quality of group descriptions. Three steps are performed: text filtering removes HTML tags, entities and non-letter characters except for sentence boundaries. Next, each snippet's language is identified and finally appropriate stemming and stop words removal end the preprocessing phase. We used stop words as potential indicators of a document's language. Other methods, such as n-gram language detection could be used alternatively. For stemming English documents we used Porter's algorithm, for Polish we employed our own simple dictionary stemmer *Lametyzator* [10].

3.2 Frequent phrase extraction

We define *frequent phrases* as recurring ordered sequences of terms appearing in the input documents. Intuitively, when writing about something, we usually repeat the subject-related keywords to keep a reader's attention. Obviously, in a good writing style it is common to use synonymy and pronouns and thus avoid annoying repetition. We believe Lingo can partially overcome the former by using the SVD-decomposed term document matrix to identify *abstract concepts*—single subjects or groups of related subjects that are cognitively different from other abstract concepts. The latter problem has not been considered in this work.

To be a candidate for a cluster label, a frequent phrase or a single term must:

1. appear in the input documents at least certain number of times (*term frequency threshold*),
2. not cross sentence boundaries,
3. be a *complete phrase* (see definition below),
4. not begin nor end with a stop word. These assumptions are discussed in detail in [2] and partially in [11].

A *complete phrase* is a complete substring of the collated text of the input documents, defined in the following way: Let T be a sequence of elements $(t_1, t_2, t_3 \dots t_n)$. S is a *complete substring* of T when S occurs in k distinct positions $p_1, p_2, p_3 \dots p_k$ in T and $\exists i, j \in 1 \dots k : t_{p_i-1} \neq t_{p_j-1}$ (left-completeness) and $\exists i, j \in 1 \dots k : t_{p_i+|S|} \neq t_{p_j+|S|}$ (right-completeness). In other words, a complete phrase cannot be "extended" by adding preceding or trailing elements, because at least one of these elements is different from the rest. An efficient algorithm for discovering complete phrases was proposed in [2], although it contained one mistake that caused the frequency of some phrases to be miscalculated. The space limits make it impossible to discuss

details here, refer to [7] for a full overview of the corrected algorithm. It does not affect further discussion of Lingo because any algorithm capable of discovering frequent phrases could be used at this stage; we use the suffix arrays approach, because it is convenient in implementation and very efficient.

3.3 Cluster label induction

Once frequent phrases (and single frequent terms) that exceed term frequency thresholds are known, they are used for cluster label induction. There are three steps to this: term-document matrix building, abstract concept discovery, phrase matching and label pruning.

The term-document matrix is constructed out of single terms that exceed a predefined term frequency threshold. Weight of each term is calculated using the standard *term frequency, inverse document frequency* (*tfidf*) formula [9], terms appearing in document titles are additionally scaled by a constant factor. In abstract concept discovery, Singular Value Decomposition method is applied to the term-document matrix to find its orthogonal basis. As discussed earlier, vectors of this basis (SVD's U matrix) supposedly represent the abstract concepts appearing in the input documents. It should be noted, however, that only the first k vectors of matrix U are used in the further phases of the algorithm (details in [7]). We estimate the value of k by selecting the Frobenius norms of the term-document matrix A and its k-rank approximation A_k. 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: $\|A_k\|_F / \|A\|_F \geq q$, where $\|X\|_F$ symbol 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 users' preferences. We therefore 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. Thus, the classic cosine distance can be used 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 and keywords 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 m_i of cosines of the angles between the i-th abstract concept vector and the phrase vectors can be calculated: $m_i = {U_i}^T P$. The phrase that corresponds to the maximum component of the m_i vector should be selected as the human-readable description of i-th abstract concept. Additionally, the value of the cosine becomes the score of the cluster label candidate. A similar process for a single abstract concept can be

extended to the entire U_k matrix—a single matrix multiplication $M = {U_k}^T P$ yields the result for all pairs of abstract concepts and frequent phrases.

On one hand we want to generalize information from separate documents, but on the other we want to make it as narrow as possible at the cluster-description level. Thus, the final step of label induction is to prune overlapping label descriptions. 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 $Z^T Z$, 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.

3.4 Cluster content discovery

In the cluster content discovery phase, the classic Vector Space Model is used to assign the input documents to the cluster labels induced in the previous phase. In a way, we re-query the input document set with all induced cluster labels. The assignment process resembles document retrieval based on the VSM model. Let us define matrix Q, in which each cluster label is represented as a column vector. Let $C = Q^T A$, where A is the original term-document matrix for input documents. This way, element c_{ij} 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 c_{ij} 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 *Others*.

3.5 Final cluster formation

Finally, clusters are sorted for display based on their score, calculated using the following simple formula: C_{score} = 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, possibly noisy ones. For the time being, no cluster merging strategy or hierarchy induction is proposed for Lingo.

4 An illustrative example

Let us assume that the following input data is given (keywords and frequent phrase extraction phase has been omitted):

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

The normalized, *tfidf*-weighted term-document matrix $\widehat{A}_{tfidf}$ is shown below together with matrix U (part of the SVD decomposition):

$$\widehat{A}_{tfidf} = \begin{vmatrix} 0 & 0 & 0.56 & 0.56 & 0 & 0 & 1 \\ 0.49 & 0.71 & 0 & 0 & 0 & 0.71 & 0 \\ 0.49 & 0.71 & 0 & 0 & 0 & 0.71 & 0 \\ 0.72 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0.83 & 0.83 & 0 & 0 & 0 \end{vmatrix} \quad U = \begin{vmatrix} 0 & 0.75 & 0 & -0.66 & 0 \\ 0.65 & 0 & -0.28 & 0 & -0.71 \\ 0.65 & 0 & -0.28 & 0 & 0.71 \\ 0.39 & 0 & 0.92 & 0 & 0 \\ 0 & 0.66 & 0 & 0.75 & 0 \end{vmatrix}$$

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: $k = 0 \mapsto q = 0.62$, $k = 1 \mapsto q = 0.856$, $\boxed{k = 2 \mapsto q = 0.959}$, so the number of clusters $k = 2$. To find descriptions of our clusters ($k = 2$ columns of matrix U), we calculate $M = {U_k}^T P$, where P is a term-document–like matrix created out of our frequent phrases and terms (values in P *tfidf*-weighted and normalized):

$$P = \begin{vmatrix} 0 & 0.56 & 1 & 0 & 0 & 0 & 0 \\ 0.71 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0.71 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0.83 & 0 & 0 & 0 & 0 & 1 \end{vmatrix} \quad M = \begin{vmatrix} 0.92 & 0 & 0 & 0.65 & 0.65 & 0.39 & 0 \\ 0 & 0.97 & 0.75 & 0 & 0 & 0 & 0.66 \end{vmatrix}$$

Rows of matrix M represent clusters, columns – their descriptions. For each row we select the column with maximum value – the two clusters are: *Singular Value* (score: 0.92) and *Information Retrieval* (score: 0.97). We skip label pruning as it is not needed in this example. Finally, documents are assigned to clusters by applying matrix Q, created out of cluster labels, back to the original matrix $\widehat{A}_{tfidf}$.

$$Q = \begin{vmatrix} 0 & 0.56 \\ 0.71 & 0 \\ 0.71 & 0 \\ 0 & 0 \\ 0 & 0.83 \end{vmatrix} \quad C = \begin{vmatrix} 0.69 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0.56 \end{vmatrix}$$

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*

5 Evaluation

Lingo has been evaluated empirically by performing an experiment on 7 users and a set of 4 search results, 2 in Polish and 2 in English. Users have been asked to establish whether cluster labels were meaningful and whether document assignments to those clusters made sense. Unfortunately, because of this paper's length limitations we are unable to present the full result of the evaluation. Full set of metrics and results is given in [7]. Let us mention here, that the results, although done on a small number of users (7), were quite promising—users found 70–80% clusters useful and 80–95% of snippets inside those clusters matching their topic. Over 75% cluster labels were marked as useful (with the distribution of noise clusters toward the lower-scoring ones).

Recently we performed another evaluation of Lingo, aimed at verifying its topic-separation capabilities. We mixed several categories from the Open Directory Project into test sets and analyzed how Lingo splits them back into original topic groups. We encountered certain problems with numerical analysis of results again, but the empirical comparison of structure of clusters created by Lingo and Suffix Tree Clustering revealed that Lingo created more concise and diverse structure of topics. More results can be found in [8], a paper published at the same conference and available in the same volume of conference proceedings as this one.

6 Conclusions and future work

We have presented a novel algorithm for clustering of Web search results. The inspiration for the algorithm was taken from both existing scientific work [2], and a commercial system—Vivisimo. Our algorithm, however, took a different path in many areas. Specifically, our contribution is in presenting a description-comes-first algorithm; to our best knowledge, no similar algorithms have been published so far. Lingo achieves impressing empirical results, but the work on the algorithm is obviously not finished. Cluster label pruning phase could be improved by adding elements of linguistic recognition of nonsensical phrases. Topic separation phase currently requires computationally expensive algebraic transformations—incremental approaches with small memory footprint would be of great importance for algorithm's scalability. It is tempting to find a method of inducing hierarchical relationships between topics. Finally, a more elaborate evaluation technique will be necessary to establish weak points in the algorithm.

Acknowledgment

The authors would like to thank an anonymous reviewer for helpful suggestions. This research has been supported by funds from an internal university grant.

References

1. Michael W. Berry, Susan T. Dumais, and Gavin W. O'Brien. Using linear algebra for intelligent information retrieval. Technical Report UT-CS-94-270, University of Tennessee, 1994.
2. Zhang Dong. *Towards Web Information Clustering*. PhD thesis, Southeast University, Nanjing, China, 2002.
3. Peter Hannappel, Reinhold Klapsing, and Gustaf Neumann. MSEEC — a multi search engine with multiple clustering. In *Proceedings of the 99 Information Resources Management Association Conference*, May 1999.

4. Marti A. Hearst and Jan O. Pedersen. Reexamining the cluster hypothesis: Scatter/gather on retrieval results. In *Proceedings of SIGIR-96, 19th ACM International Conference on Research and Development in Information Retrieval*, pages 76–84, Zürich, CH, 1996.
5. Udi Manber and Gene Myers. Suffix arrays: a new method for on-line string searches. In *Proceedings of the first annual ACM-SIAM symposium on Discrete algorithms*, pages 319–327, 1990.
6. Irmina Masłowska. Phrase-Based Hierarchical Clustering of Web Search Results. In *Proceedings of the 25th European Conference on IR Research, ECIR 2003*, volume 2633 of *Lecture Notes in Computer Science*, pages 555–562, Pisa, Italy, 2003. Springer.
7. Stanisław Osiński. An Algorithm for Clustering of Web Search Results. Master's thesis, Poznań University of Technology, Poland, 2003. [@:] `http://www.cs.put.poznan.pl/dweiss/carrot-bin/osinski-2003-lingo.pdf`.
8. Stanisław Osiński and Dawid Weiss. Conceptual clustering using Lingo algorithm: Evaluation on Open Directory Project data. Submitted to Intelligent Information Systems Conference 2004, Zakopane, Poland, 2003.
9. Gerard Salton. *Automatic Text Processing — The Transformation, Analysis, and Retrieval of Information by Computer*. Addison–Wesley, 1989.
10. Dawid Weiss and Jerzy Stefanowski. Web search results clustering in Polish: Experimental evaluation of Carrot. In *Proceedings of the New Trends in Intelligent Information Processing and Web Mining Conference, Zakopane, Poland*, 2003.
11. Oren Zamir and Oren Etzioni. Grouper: a dynamic clustering interface to Web search results. *Computer Networks (Amsterdam, Netherlands: 1999)*, 31(11–16):1361–1374, 1999.