Tag Cloud Control
by Latent Semantic Analysis

submitted by
Angelina Velinska

supervised by
Prof. Dr. Ralf Möller
Dipl. Ing. Sylvia Melzer
Software Systems Institute (STS)
Technical University of Hamburg-Harburg

Dr. Michael Fritsch
CoreMedia AG
Hamburg

Declaration

I declare that:
this work has been prepared by myself,
all literal or content based quotations are clearly pointed out,
and no other sources or aids than the declared ones have been used.

Hamburg, December 2010
Angelina Velinska

Acknowledgements

TO BE DONE

Contents

List of Figures

Chapter 1

Latent Semantic Analysis

Summary. *The chapter gives a theoretical overview of Latent Semantic Analysis (LSA) in the context of its use in this work.*

1.1 Overview

LSA was first introduced in [1] and [2] as a technique for improving information retrieval. Most search engines work by matching words in a user's query with words in documents. Such information retrieval systems that depend on lexical matching have to deal with two problems: synonymy and polysemy. Due to the many meanings which the same word can have, also called polysemy, irrelevant information is retrieved when searching. And as there are different ways to describe the same concept, or synonymy, important information can be missed. LSA has been proposed to address these fundamental retrieval problems, having as a key idea dimension reduction technique, which maps documents and terms into a lower dimensional semantic space. LSA models the relationships among documents based on their constituent words, and the relationships between words based on their occurrence in documents. By using fewer dimensions that there are unique words, LSA induces similarities among words including ones that have never occurred together [3]. There are three basic steps to using LSA: text pre-processing, computing Singular Value Decomposition (SVD) and dimensionality reduction, and querying the constructed semantic space.

1.2 Text pre-processing

If we have a document collection or a text corpus, on which we want to apply LSA, the initial step is to pre-process the texts into a suitable form for running LSA. Pre-processing can include a number of techniques, depending on the application requirements. The process of parsing, also called tokenization, is breaking the input text stream into useable tokens. During tokenization, filtering can be applied, i.e. removing HTML tags or other markup, as well as stop-wording, and removing punctuation marks. Stop words don't convey information specific to the text corpus, but occur frequently, such as: *a*, *an*, *and*, *any*, *some*, *that*, *this*, *to*.

A distinction has to be made between words or terms, and tokens. A term is the class which is used as a unit during parsing, and a token is each occurence of this class. For example, in the sentence:

> *CoreMedia CMS is shipped with an installation program for interactive graphical installation and configuration of the software.*

the term *installation* is represented by two tokens.

There is no universal way in which to parse a text, and the parsing decisions to address depend on the application in which the text collection will be used. Text parsing will influence all posterior processing in the following stages of LSA.

After tokenization, one has to construct a term-document matrix (1.1). Having as rows the terms, and as columns the documents, its elements are the occurrences of each term in a particular document, where a_{ij} denotes the frequency with which term i occurs in document j. The size of the matrix is **m x n**, where **m** is the number of terms, and **n** is the number of documents in the text collection. Since every term doesn't appear in each document, the matrix is usually sparse.

$$A = \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} \tag{1.1}$$

Local and global weightings are applied to increase or decrease the importance of terms within documents. We can write

$$a_{ij} = L(i,j) \times G(i), \tag{1.2}$$

where $L(i, j)$ is the local weighting of the term i in document j, and $G(i)$ is the global weighting for term i. The choice of a weight function has impact on LSA performance, therefore in Section 1.5 we give an overview of the most common weight functions.

1.3 Singular Value Decomposition

After the initial pre-processing, the term-document matrix is decomposed into three matrices (1.3) by applying Singular Value Decomposition (SVD). It is a unique decomposition of a matrix into the product of three matrices - U and V are ortonormal matrices, and Σ is a diagonal matrix having singular values on its diagonal.

$$A = U\Sigma V^T \tag{1.3}$$

After the initial matrix A is decomposed, all but the highest k valued of S are set to 0. The resulting reduced matrix is the semantic space of the text collection. A classical example presenting the truncated SVD [1] can be used for displaying dimensionality reduction, and how it affects all three matrices.

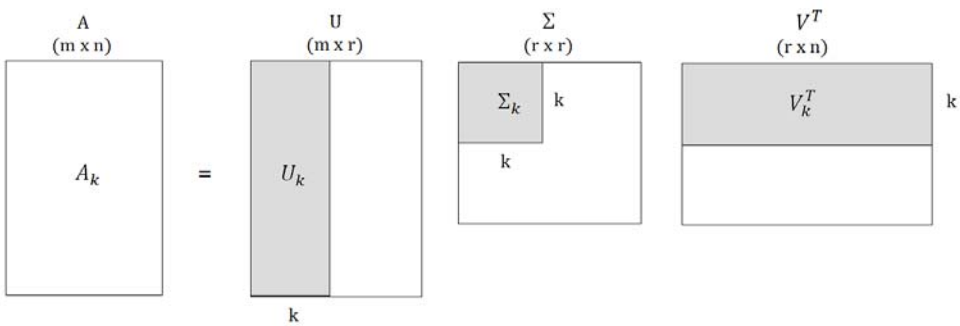

Figure 1.1: Diagram of truncated SVD

A_k - best rank-k approximation of A	m - number of terms
U - term vectors	n - number of documents
Σ - singular values	k - number of factors
V^T - document vectors	r - rank of A

Figure 1.1 is a visual representation of SVD as defined in equation (1.3). U and V are considered as containing the term and document vectors

respectively, and Σ is constructed by the singular values of A. An imporant property of SVD is that the singular values placed on the diagonal of Σ are in decreasing order. Hence, if all but the first k singular values are set to 0, the semantic meaning in the resulting space is preserved to some approximation k, while noise or variability in word usage, is filtered out. Noise in this case are the terms with lowest weights which carry little meaning. By using fewer dimensions k, LSA induces similarities amont terms including ones that have never occured together. Terms which occur in similar documents, for example, will be near each other in the k-dimensional space even if they never co-occur in the same document. This means that some documents which do not share any words with a users query may be near it in k-space.

A factor to be considered when computing SVD is the run-time complexity of the algorithm. For decomposition of very large matrices, it is $O(n^2k^3)$, where n is the number of terms in the text corpus, and k is the number of dimensions in semantic space after dimensionality reduction. Note that k is typically a small number between 50 and 350.

A more detailed description of SVD can be found in [4] and [5].

1.4 Querying the semantic space

In this work we are using LSA for Information Retrieval (IR) purpose. Therefore, the final step of applying the technique is to pose queries on the constructed semantic space. A query q is a set of words which must be represented as a document in the k-dimensional space, in order to be compared to other documents. The user's query can be represented by

$$q = q^T U_k \Sigma_k^{-1} \tag{1.4}$$

where q is the set of words in the query, multiplied by the reduced term and singular values matrices. Using the transformation in (1.4), the query is "mapped" onto the reduced k-space. After the mapping, the resulting query vector can be compared to the documents in the k-space, and the results ranked by their similarity or nearness to the query. A common similarity measure is the cosine between the query and the document vector. From the resulting document set, the documents closest to the query above certain treshold are returned.

1.5 Factors influencing LSA performance

The effective usage of LSA is a process of a sophisticated tuning. Several factors can influence the performance of the technique. These factors are pre-processing of texts (removal of stop-words, filtering, stemming), frequency matrix transformations, choice of dimensionality k, choice of similarity measure.

Dumais et al. [6] and Nakov et al. [7] have carried research on LSA performance depending on the choice of factors such as frequency matrix transformations, similarity measures, and choice of dimension reduction parameter k. They conclude that performance based on the choice of these factors depends on the particular text corpus, as well as on the purpose of LSA application. However, in the case of matrix transform, log-entropy performs better as compared to other matrix transform function combinations, including the popular term frequency - inverse document frequency ($tf \times idf$). Therefore, we implement the former in this work.

Local function: logarithm	$L(i,j) = \log(tf(i,j)+1)$
Global function: entropy	$G(i) = 1 + \frac{\Sigma_j p(i,j)}{\log n}$

where n is the number of documents in the collection.

Further, it has been stated ([6],[8]) that with respect to similarity measures used, LSA performs optimal when cosine similarity measure is implemented to calculate the distance between vectors in the semantic space. We have therefore used it to measure the relevance between queries and documents. The cosine measure between two vectors d_1 and d_2 is given by:

$$sim(d1, d2) = \frac{\overrightarrow{V}(d_1).\overrightarrow{V}(d_2)}{\left|\overrightarrow{V}(d_1)\right| . \left|\overrightarrow{V}(d_2)\right|} \tag{1.5}$$

Dimensionality reduction parameter k is defined empirically based on the experimentation results presented in Chapter **??**.

Acronyms

CMS Content Management Systems.

IR Information Retrieval.

LSA Latent Semantic Analysis.

SVD Singular Value Decomposition.

Appendix A

Appendix

TODO: insert important source code parts here.

Bibliography

[1] S. T. Dumais, G. W. Furnas, T. K. Landauer, S. Deerwester, and R. Harshman, "Using Latent Semantic Analysis to improve access to textual information," in *Sigchi Conference on Human Factors in Computing Systems*, pp. 281–285, ACM, 1988.

[2] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman, "Indexing by Latent Semantic Analysis," *Journal of the American Society for Information Science*, vol. 41, pp. 391–407, 1990.

[3] S. Dumais, "LSA and Information Retrieval: Getting Back to Basics," pp. 293–321, 2007.

[4] M. W. Berry, S. Dumais, G. O'Brien, M. W. Berry, S. T. Dumais, and Gavin, "Using linear algebra for intelligent information retrieval," *SIAM Review*, vol. 37, pp. 573–595, 1995.

[5] G. H. Golub and C. F. Van Loan, *Matrix computations (3rd ed.)*. Baltimore, MD, USA: Johns Hopkins University Press, 1996.

[6] S. T. Dumais, "Improving the retrieval of information from external sources," *Behavior Research Methods, Instruments, & Computers*, vol. 23, pp. 229–236, 1991.

[7] P. Nakov, A. Popova, and P. Mateev, "Weight functions impact on LSA performance," in *EuroConference RANLP'2001 (Recent Advances in NLP*, pp. 187–193, 2001.

[8] P. Nakov, "Getting better results with Latent Semantic Indexing," in *In Proceedings of the Students Prenetations at ESSLLI-2000*, pp. 156–166, 2000.