MAI4CAREU Master programmes in Artificial Intelligence 4 Careers in Europe

University of Ruse

Information Retrieval

Yordan Kalmukov

May 2023

Co-financed by the European Union
Connecting Europe Facility

This Master is run under the context of Action No 2020-EU-IA-0087, co-financed by the EU CEF Telecom under GA nr. INEA/CEF/ICT/A2020/2267423

Content-based document retrieval

Latent Semantic Analysis

Latent semantic analysis (LSA), also known as latent semantic indexing (LSI) [1] is a method for reducing the dimensionality of the vectors describing the documents in a given collection that detects the relationships (similarities) not only between individual documents, but also between the words they contain. The latter is its main difference with the much easier to implement vector space model (VSM) for text analysis. The presumption is that terms that have a similar meaning often occur together in the same documents. There is logic in this, because when a person writes a text he tries to avoid repeated duplication of words, as a result of which he often replaces them with synonyms or other means of expression having a similar meaning. In this sense, LSA can identify and subsequently group semantically related words into more general "topics".

The method is called latent semantic analysis, since the topics in question are actually hidden (latent), i.e. are not explicitly stated or named, but the method "discovers" them by analyzing the semantically related words it finds and groups. Discovered topics essentially represent concepts to which, however, the method cannot give any specific title in natural language. I.e. at the algorithmic level the concepts remain untitled, but if a person who interprets them decides – he/she can always give them some name. *Each unique word in the collection of documents is related to a certain degree (a little or a lot, or maybe not at all) with a topic, and in turn, each topic has a certain contribution (share) in the description of each of the documents.* In the VSM, the meaning of a document is derived from the words it contains. In LSA, the meaning of a document is extracted from the discovered latent topics, which are obtained by grouping the semantically related words together. I.e. *latent topics represent an intermediate, implicit (hidden) level of document description*, where documents are described not by the words themselves, but by the discovered topics. Since the discovered significant latent topics within the document collection are many times less than the number of unique words, hence the reduction in the dimensionality of the vectors describing the documents.

For example, the words: space, accelerator, shuttle, rocket, probe, and planet form a "space topic." A document is related to the space topic if it contains any one or more of these words. Thus, two or more documents can be identified as semantically related, even if they do not contain any words in common. In comparison, in VSM, each word is treated as independent of the others, and if one document talks about rockets and another one about shuttles or interplanetary probes, then the semantic similarity between these documents will be zero. But not with LSA. In this sense, latent semantic analysis reliably deals with synonyms and partly with polysemy, which is its biggest advantage over the vector space model. Since words with similar meanings are grouped together into common topics, the pre-processing of the text can be greatly simplified by omitting the stemming of the words. In theory it is not necessary, because with or without stemming, the corresponding words will enter the same topic, whether as one or more words. Of course, this assumption will be experimentally tested later.

As input data, LSA takes the *term-document matrix* - the leftmost part of figure 1. In it, words are arranged in rows and documents in columns. In general, it shows how many times each word occurs in each document, but the values in the matrix can also be the tf-idf weights of the words with respect to individual documents. My experimental studies show that if the tf-idf weights are used in the matrix instead of the raw frequencies of occurrence (the number of occurrences of the words in the documents), the calculated similarities between individual documents or between the query and the documents are significantly more accurate.

Let the term-documents matrix be called A. Then row a_i contains the weights of the i-th term with respect to all documents. Similarly, row a_p contains the weights of the p-th term with respect to the same documents. Then the scalar product between the two rows (vectors) $a_i^T a_p$ will give a scalar indicating how related these two terms *i* and *p* are. If cosine normalization is applied to it, then the similarity coefficient between the terms will be obtained. It is in this way that LSA determines the relationships between individual words in the document collection. Similarly, computing the matrix product AA^T will yield a matrix containing the similarity coefficients between all combinations of words in the entire collection of documents. Similarly, the product $A^T A$ will give the similarities between all the documents in the collection.

Calculating similarities between individual words is important, but in no way groups the semantically related ones and does not lead to the discovery of the latent topics. The latter is performed using a mathematical operation called *singular value decomposition (SVD)*. According to this operation, any rectangular matrix *A* can be decomposed into the following matrix product:

$$A = U\Sigma V^T \qquad (1)$$

where U and V are orthogonal and Σ is a diagonal matrix, i.e. all its values outside the main diagonal are zero. The values of Σ are called *singular values*. They are arranged in descending order along the main diagonal and indicate the level of significance of the discovered latent topics. The values of U are called *left singular values* and *indicate the contribution of each word to each latent topic*. The values of V^T are called *right singular values* and *indicate the participation (contribution) of each discovered topic in each document* in the collection. The idea is presented in fig. 1 with 4 words, 3 documents (A_{4x3}) and 2 latent topics.

Term-document matrix

	doc 1	doc 2	doc 3
term 1	…	…	…
term 2	…	…	…
term 3	…	…	…
term 4	…	…	…

=

Term to topic contribution

	topic 1	topic 2
term 1	…	…
term 2	…	…
term 3	…	…
term 4	…	…

x

Topic significance

	topic 1	topic 2
topic 1	…	0
topic 2	0	…

x

Topic to document contribution

	doc 1	doc 2	doc 3
topic 1	…	…	…
topic 2	…	…	…

Figure 1. SVD decomposition of the term-document matrix into three matrices, providing the significance of each latent topic and the contribution of terms to topics and topics to documents.

It should be noted here that if A has a dimension of $m_x n$, then dimension of U is $m_x m$, dimension of Σ is $m_x n$, and the dimension of V^T is $n_x n$. But since LSA is a dimensionality reduction technique as well, then only the highest *k* number of singular values could be taken into account, i.e. the most significant *k* number of latent topics, and their corresponding eigenvectors from U and V. In this way we implement the so-called *truncated SVD*. Thus, the dimensionality of U becomes $m_x k$, of Σ - $k_x k$, and of V^T - $k_x n$. Of course, after performing the truncated SVD, the three matrices represent a *sufficiently accurate approximation, but not an identical representation of the original matrix*.

It is known from linear algebra that the columns of U are actually *the eigenvectors* of the matrix product AA^T, the columns of V (or the rows of V^T) are the *eigenvectors* of the product A^TA, and the values of Σ are square root of the *eigenvalues* of AA^T or A^TA. In other words, in order to implement the singular value decomposition, the eigenvalues and the eigenvectors of the matrix products AA^T and A^TA must be computed.

The following equation is also known from the linear algebra:

$$Av = \lambda v \tag{2}$$

where A is a matrix, v is an *eigenvector* and λ – *eigenvalue* of the matrix A.

Equation (2) could be rewritten in the form of (3)

$$(A - \lambda E)v = 0 \tag{3}$$

where E is the identity matrix (In English literature it is noted as I).

Equation (3) could have a non-zero eigenvector v, only if the determinant of the matrix $(A - \lambda E)$ is zero. I.e.

$$|A - \lambda E| = 0 \text{ or } \det(A - \lambda E) = 0 \tag{4}$$

Equation (4) is called *characteristic polynomial* of the matrix A. It represents a polynomial of λ of the n-th degree, where *n* is the rank of the matrix. This means that it has at most *n* number of different solutions for λ, i.e. *n* of number of eigenvalues. Each eigenvalue λ corresponds to a nonzero eigenvector v. In fact, when solving the system of linear equations (3) by substitution of the eigenvalue, multiple eigenvectors corresponding to the same eigenvalue can be obtained. However, all these vectors are in the same direction, therefore only one corresponding eigenvector is taken for an eigenvalue, such that *the eigenvectors* of the matrix products AA^T and A^TA *are orthonormal*.

As already mentioned in order to implement the singular value decomposition (SVD) of the matrix A, one must calculate the eigenvalues and the eigenvectors of the matrix products AA^T and A^TA. This is done by calculation of the determinant (4), which is, however, a computationally intensive task, especially for high-order determinants.

There are different ways to calculate determinants of matrices. Some are studied in higher mathematics courses – for example, method of adjoint quantities (Laplace expansion), Gaussian elimination, Gauss–Jordan elimination, etc. The method of adjoint quantities is suitable for demonstration and teaching purposes because it is easy to understand, but its time complexity of O(n!) makes it completely inapplicable to practical calculus of high-order determinants. Gaussian and Gauss-Jordan elimination have significantly better time complexity of O(n^3), and their modifications form the basis of some of the most widely used algorithms and methods for computing determinants. For example, the LU decomposition (lower–upper decomposition) proposed by Tadeusz Banachiewicz is based on Gaussian elimination [2]. The algorithm of Erwin Bareiss [3] is also a variant of Gaussian elimination. Both have time complexity of O(n^3). The fastest known algorithm for calculating determinant is the Fast matrix multiplication of James Bunch and John Hopcroft [4,5] with complexity O($n^{2.373}$).

There are also algorithms that directly calculate the eigenvalues and eigenvectors. For example, the QR algorithm [6,7,8], proposed in the early 1960s by John Francis and Vera Kublanovskaya, or the algorithm named after Carl Gustav Jacobi [9], which proposed a similar method as early as 1846. For singular value decomposition (SVD) of matrices, the version of the QR algorithm proposed by Gene H. Golub and William Kahan [10] in 1965 is commonly used.

After finding the eigenvalues and eigenvectors of the matrix products AA^T and A^TA: square root of the eigenvalues, ordered by magnitude, form the main diagonal of Σ; the eigenvectors of AA^T, positionally mapped to the ordered eigenvalues, form the columns of U; and the eigenvectors of A^TA, again positionally mapped to the eigenvalues—the columns of V (or the rows of V^T). To implement a truncated SVD decomposition, in the matrices U, Σ and V only those rows or columns remain that correspond to the first *k* highest, singular values, i.e. of the first *k* number of most significant latent topics. Of course, the truncated SVD decomposition no longer represents an exact decomposition of the original matrix A. The resulting matrices U_k, Σ_k and V_k^T correspond to the singular value decomposed matrix A_k, which is a rank *k* approximation of the original A. Quite naturally, with this approximation, some, albeit minimal, error occurs. However, the reduction in the dimensionality and the transition from a *word space* to a more semantically oriented *topic space* is achieved in this way.

Finally, to calculate the *degree of similarity between any two documents* in the collection, based on the reduced number of latent topics, one should simply calculate the cosine of the angle between the corresponding two columns in V_k^T. If it is necessary to calculate a similarity between a query *q* and one of the documents in the collection, given that *q* is not present in V_k^T, then first *q* must be transformed into a k-dimensional vector q_k (considering only the most significant *k* number of topics) to have the same dimension as the columns of V_k^T. The similarity is then calculated again by the cosine of the angle between q_k and the corresponding column in V_k^T. The transformation of *q* into q_k is done by the matrix equation (5).

$$q_k = \Sigma_k^{-1} U_k^T q \qquad (5)$$

One of the main goals of latent semantic analysis is to calculate the degree of similarity between two documents or between a query and a document. However, the accuracy of the calculated similarity is influenced by a number of factors, the most significant of which are:

- The number of latent (hidden) topics.
- The way values in the term-document matrix are calculated. If they are simply frequencies of occurrence of individual words in the relevant documents or tf-idf weights.
- The models for calculating term weights if the values in the matrix are tf-idf weights.
- Whether the words are pre-stemmed or not.

The number of used latent topics significantly affects the accuracy of the calculated similarities, but unfortunately, there is no theoretically motivated rule by which this number can be fixed in advance. It depends strongly on other factors, such as the number of documents in the collection, the number of unique words (terms) in the dictionary, the way term weights are calculated, etc. The most accurate way to determine the appropriate number of latent topics for a given task or collection of documents is experimentally, which is somewhat of a problem. Using too many topics will make Latent Semantic Analysis (LSA) behave like the Vector Space Model (VSM), treating words as separate and independent. On the other hand, using too small a number of latent topics will cause LSA to start grouping semantically unrelated words into common topics, which will also lead to a loss of accuracy. According to the experimental data of scientists, for a collection of about 5000 documents and a matrix containing only the frequency of occurrence of individual words in the corresponding documents, a value of 100 is a good starting point to experimentally determine the optimal number of latent topics. Logically, if the number of unique words and documents in the collection increases/decreases, the optimal number of latent topics should also increase/decrease. Own experimental studies fully confirm this assumption.

In general, latent semantic analysis is performed on a term-document matrix composed of the number of occurrences of each word in the corresponding documents. But its elements can also be tf-idf weights. Results from my experiments show that the calculated similarities are about 5-7 percentage points (ppts) higher if the matrix is composed of tf-idf weights. As opposed to VSM, Latent Semantic Analysis achieves higher accuracy of similarities if IDF is applied to both query terms and document terms.

As already stated, word stemming should not have a significant impact on the accuracy of calculated similarities, since LSA groups semantically related words into common topics. Whether multiple derivatives of one word will be grouped up in one topic, or multiple stemmed words will be recognized as one word, is pretty much the same. Almost, but not quite, because in the first case the many words although grouped still remain many words, while in stemming they are recognized as one. And if other semantically related words are grouped with them, then small nuances would appear - but really small ones that would not significantly change the calculated similarities.

Regarding the accuracy of the calculated similarities, detailed experimental studies and comparison with other considered methods for document description and text analysis have been carried out. LSA is expected to perform better than VSM, mainly due to its ability to deal with synonyms and clustering of semantically related words together. A number of authors report experimental studies, and comparative analyzes of LSA and VSM applied to the standardized datasets MED, CISI, NPL, TIME and CACM, which are used to evaluate different document search and retrieval methods. For most of them, the LSA and VSM results are comparable. However, on the MED collection, Latent Semantic Analysis achieves about 15 percentage points (about 30%) higher average accuracy than VSM. For CISI and NPL, the improvement is about 2 percentage points, which is, however, a 10% difference because both methods achieve quite low precision in these collections. For the TIME and CACM collections, LSA performs even worse than VSM at numbers of latent topics below 400 for TIME and 2500 CACM, after which the precision of the two methods level off. The authors discuss what could be the reasons for these very heterogeneous results. Moldovan [11] and Li [12] concluded that both the structure of the standardized sets and the data itself (the subject area) probably influence the accuracy of the methods. It should be noted here that standardized data sets are a kind of "lab data" that contain not only the raw data, but also pre-prepared queries and a list of results for each query, indicating which result is relevant and which is not. However, the results rating is often provided by one expert only, just once and in a specific context. In a study by Dumais and Nielsen [13], an expert was asked to repeat his rating about the relevance of the results (the same results) about 7 months after his first survey, and it turned out that his second rating differed significantly from the first, with the correlation between the two answers/ratings of 0.76.

Andreea Moldovan et al [11] conducted an experimental comparative analysis between LSA and VSM applied to a collection of patent documents from the US Patent Office for the period 1790 to 2005. Their analysis shows that the results of LSA and VSM are actually quite close, as in most cases, LSA does perform better, but the improvement is about 5%. However, the authors also cite cases where LSA performs worse than VSM, achieving 3% lower document retrieval accuracy than VSM.

Experimental studies and comparative analyzes conducted by me also show that the Latent Semantic Analysis achieves higher accuracy of the calculated similarities than the Vector Space Model. When the term-document matrix is formed only from the frequency of occurrence of the individual words in the corresponding documents, the increase in accuracy is only 2-3 percentage points (ppts), while if the matrix consists of the tf-idf term weights, then the accuracy rises on average by about 8-9 percentage points (which is actually over 10%). It is important to note here that the comparison with VSM is done when using the best term weighting model for VSM - BM 25. Compared to the base tf-idf model, the increase is just over 10 percentage points.

The Latent Semantic Analysis is a reliable method for calculating similarities between individual documents or between a query and documents in a collection. However, before it is preferred over other methods, it is desirable to analyze the specific task and possibly consider whether it could be replaced by the much easier to implement Vector Space Model for text analysis. LSA's time complexity of O(n^3), given the expected huge values of *n* in the tens of thousands, makes it not very suitable for real-time work on large document collections. For a collection of about 5,000 documents, the number of unique words in it is about and above 20,000, which also determines the value of *n*. However, unlike VSM, there is no possibility of using an inverted index with LSA.

References:

1. Deerwester S., Dumais S.T., Furnas G.W., Landauer T.K. and Harshman R. (1990): Indexing by latent semantic analysis. —J. Amer. Soc. Inf. Sci., Vol. 41, No. 6, pp. 391–407.
2. Schwarzenberg-Czerny, A. (1995). "On matrix factorization and efficient least squares solution". Astronomy and Astrophysics Supplement Series. 110: 405.
3. Bareiss, Erwin H. (1968), "Sylvester's Identity and multistep integer-preserving Gaussian elimination", Mathematics of Computation, 22 (103): 565–578, doi:10.2307/2004533, JSTOR 2004533
4. Bunch, J. R.; Hopcroft, J. E. (1974). "Triangular Factorization and Inversion by Fast Matrix Multiplication". Mathematics of Computation. 28 (125): 231–236. doi:10.1090/S0025-5718-1974-0331751-8.
5. Aho, Alfred V., Hopcroft, John E., Ullman, Jeffrey D. (1974), The Design and Analysis of Computer Algorithms, Addison-Wesley, Theorem 6.6, p. 241

6. Francis, J.G.F. "The QR Transformation, I", The Computer Journal, 4(3), pages 265–271 (1961). doi:10.1093/comjnl/4.3.265
7. Francis, J. G. F. (1962). "The QR Transformation, II". The Computer Journal. 4 (4): 332–345. doi:10.1093/comjnl/4.4.332
8. Vera N. Kublanovskaya, "On some algorithms for the solution of the complete eigenvalue problem," USSR Computational Mathematics and Mathematical Physics, vol. 1, no. 3, pages 637–657 (1963). doi:10.1016/0041-5553(63)90168-X
9. Golub, G.H.; van der Vorst, H.A. (2000). "Eigenvalue computation in the 20th century". Journal of Computational and Applied Mathematics. 123 (1–2): 35–65. doi:10.1016/S0377-0427(00)00413-1
10. Golub, Gene H.; Kahan, William (1965). "Calculating the singular values and pseudo-inverse of a matrix". Journal of the Society for Industrial and Applied Mathematics, Series B: Numerical Analysis. 2 (2): 205–224. Bibcode:1965SJNA....2..205G. doi:10.1137/0702016. JSTOR 2949777.
11. Moldovan, Andreea, Radu Ioan Bot, and Gert Wanka. "Latent semantic indexing for patent documents." (2005).
12. Li, Dandan, and Chung-Ping Kwong. "Understanding latent semantic indexing: A topological structure analysis using Q-analysis." Journal of the american society for information science and technology 61, no. 3 (2010): 592-608.
13. Dumais, Susan T., and Jakob Nielsen. "Automating the assignment of submitted manuscripts to reviewers." In Proceedings of the 15th annual international ACM SIGIR conference on Research and development in information retrieval, pp. 233-244. 1992.