Latent Semantic Analysis

Zhu Zhiyuan

July 18th, 2017

Introduction

Latent semantic analysis (LSA) is a method for exploring the underlying meaning of words. LSA suppose that words that are close in meaning will occur in similar part of text, which is that ”a word is characterized by the company it keeps"**[1]** Linear algebra methods, in particular, singular value decomposition (SVD), is used to solve particular problems.

Moreover , LSA has been shown to reflect human knowledge. In addition, LSA has found application in a number of areas, including selecting educational materials for individual students and improving reading comprehension of students.

LSA

Latent Semantic Analysis (LSA) is a method for discovering underlying meaning in document. **[2]** In a given file , each document and term is expressed as a vector. Each element in a vector shows the frequency of the document or term context . In this way,

document-document, document-term, and term-term similarities or semantic relationship can be unveiled by machine.

Here is an simple Example of how LSA works in a particular situation **[2]**

“ *Suppose we have the following set of five documents d1 : Romeo and Juliet.*
d2 : Juliet: O happy dagger!
d3 : Romeo died by dagger.
d4 : “Live free or die”, that’s the New-Hampshire’s motto.
d5 : Did you know, New-Hampshire is in New-England.
and a search query: dies, dagger.
Clearly, d3 should be ranked top of the list since it contains both dies, dagger. Then, d2 and d4
should follow, each containing a word of the query. However, what about d1 and d5? Should they be returned as possibly interesting results to this query? As humans we know that d1 is quite related to the query. On the other hand, d5 is not so much related to the query. Thus, we would like d1 but not d5, or differently said, we want d1 to be ranked higher than d5.
The question is: Can the machine deduce this? The answer is yes, LSI does exactly that. In this example, LSI will be able to see that term dagger is related to d1 because it occurs together with the d1’s terms Romeo and Juliet, in d2 and d3, respectively. Also, term dies is related to d1 and d5 because it occurs together with the d1’s term Romeo and d5’s term New-Hampshire in d3 and d4, respectively. LSI will also weigh properly the discovered connections; d1 more is related to the query than d5 since d1 is “doubly”

connected to dagger through Romeo and Juliet, and also connected to die through Romeo, whereas d5 has only a single connection to the query through New-Hampshire. ”

Mathematical Foundations of LSA

How latent semantic analysis (LSA) works on computer is based on the vector space models. By using vector space model, LSA can retrieval text from a super huge information database effectively and automatically.

When the vector space model for latent semantic analysis is created, singular value decomposition (SVD) is used to decomposed the vector space. It is by using the SVD that LSA obtains the meanings of types and documents.

Create the Vector Space Model

A vector space model for latent semantic analysis (LSA) is a type-by-document matrix. **[3]**The rows of the input matrix are made up of types, which are the individual components that make up a document. The columns of the input matrix are made up of documents. Therefore, a document that has *n* documents and *m* types can be

transformed into an *m* by *n* type-by-document matrix **A**.Each column of the matrix A contains zero and nonzero elements, aij, who has a frequency of ith type in the jth document.

TABLE 1

Titles for Topics on Music and Baking

Label	*Titles*
M1	*Rock* and *Roll Music* in the 1960's
M2	Different *Drum Rolls*, a Demonstration of Techniques
M3	*Drum* and Bass *Composition*
M4	A Perspective of *Rock Music* in the 90's
M5	*Music* and *Composition* of Popular Bands
B1	How to Make *Bread* and *Rolls*, a *Demonstration*
B2	*Ingredients* for Crescent *Rolls*
B3	A *Recipe* for Sour*dough Bread*
B4	A Quick *Recipe* for Pizza *Dough* using Organic *Ingredients*

Note. Keywords are in italics.

TABLE 2

Types	*Documents*								
	M1	M2	M3	M4	M5	B1	B2	B3	B4
Bread	0	0	0	0	0	1	0	1	0
Composition	0	0	1	0	1	0	0	0	0
Demonstration	0	1	0	0	0	1	0	0	0
Dough	0	0	0	0	0	0	0	1	1
Drum	0	1	1	0	0	0	0	0	0
Ingredients	0	0	0	0	0	0	1	0	1
Music	1	0	0	1	1	0	0	0	0
Recipe	0	0	0	0	0	0	0	1	1
Rock	1	0	0	1	0	0	0	0	0
Roll	1	1	0	0	0	1	1	0	0

Here is an example above **[3]**.

" *A small example of a document collection and its corresponding input type-by-document matrix with type frequencies can found in above [BB}. In this example, documents are the actual title, consisting only of italicized keywords. Documents labeled M1–M5 are music-related titles, documents labeled B1–B4 are baking-related titles, and no document has more than one occurrence of a type or keyword.* "

Also, a weighting is applied to the each nonzero element**,** in am attempt to improve text retrieval **[4]**. When retrieving paper , the types that best distinguish documents from the rest are the most important. Therefore, the computer should give a low weight to a high-frequency type that occurs in many documents and a high weight to types that occur in some documents but not all **[5]**.

Also, LSA utilities a local and global weighting to each nonzero element, in order to further distinguish the importance of types within documents (local) and inter-document collection (global). The local and global weighting functions for each element are usually directly related to how frequently a type occurs within a document and inversely related to how frequently a type occurs in documents across the collection, respectively.

[6]

" Local weighting functions include using type frequency, binary frequency (0 if the type is not in the document and 1 if the type is in the document), and log of type frequency plus 1. Global weighting functions include normal, gfidf, idf, and entropy, all of which basically assign a low weight to types occurring often or in many documents. A common local and global weighting function is log-entropy. Dumais found that log-entropy gave the best retrieval results, 40% over raw type frequency (Dumais, 1991). The local weighting function of log (type frequency + 1) decreases the effect of large differences in frequencies.

Entropy, defined as 1 + $\sum_j \frac{p_{ij} \log_2(p_{ij})}{\log_2 n}$

where $p_{ij} = \frac{tf_{ij}}{gf_i}$

tf_{ij} *= type frequency of type i in document j,*

and gf_i *= the total number of times that type i appears in the entire collection of n documents , gives less weight to types occurring frequently in a document collection, as well as taking into account the distribution of types over documents (Dumais, 1991). Below table has the local and global weighting function log-entropy applied to each nonzero type frequency in the type-by-document matrix given previously in previous one. "*

TABLE 3

Types	*Documents*								
	M1	M2	M3	M4	M5	B1	B2	B3	B4
Bread	0	0	0	0	0	.474	0	.474	0
Composition	0	0	.474	0	.474	0	0	0	0
Demonstration	0	.474	0	0	0	.474	0	0	0
Dough	0	0	0	0	0	0	0	.474	.474
Drum	0	.474	.474	0	0	0	0	0	0
Ingredients	0	0	0	0	0	0	.474	0	.474
Music	.347	0	0	.347	.347	0	0	0	0
Recipe	0	0	0	0	0	0	0	.474	.474
Rock	.474	0	0	.474	0	0	0	0	0
Roll	.256	.256	0	0	0	.256	.256	0	0

Decomposition of Input Matrix Into Orthogonal Components

Once the input matrix **A** is created, it is transformed into a type and document vector space by orthogonal decompositions in order to exploit truncation of the vectors. Properties of the matrix are preserved by transforming a matrix by using orthogonal decompositions .

" *An orthogonal matrix is one with the property of* $\boldsymbol{Q}^T\boldsymbol{Q} = \boldsymbol{I}$*, where* $\boldsymbol{Q}$ *is an orthogonal matrix,* $\boldsymbol{Q}^T$ *is the transpose of matrix* $\boldsymbol{Q}$ *(the rows and columns of* $\boldsymbol{Q}$ *are the columns and rows of* $\boldsymbol{Q}^T$*), and* $\boldsymbol{I}$ *is the identity matrix: "* **[6]**

$$Q^T Q = \begin{bmatrix} 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & 0 & & 0 \\ \vdots & 0 & \ddots & \ddots & \vdots \\ 0 & & \ddots & 1 & 0 \\ 0 & 0 & \dots & 0 & 1 \end{bmatrix}$$

LSA decompose the type-by-document input matrix **A** by using the singular value decomposition (SVD). There are several significant advantages of using SVD in LSA.

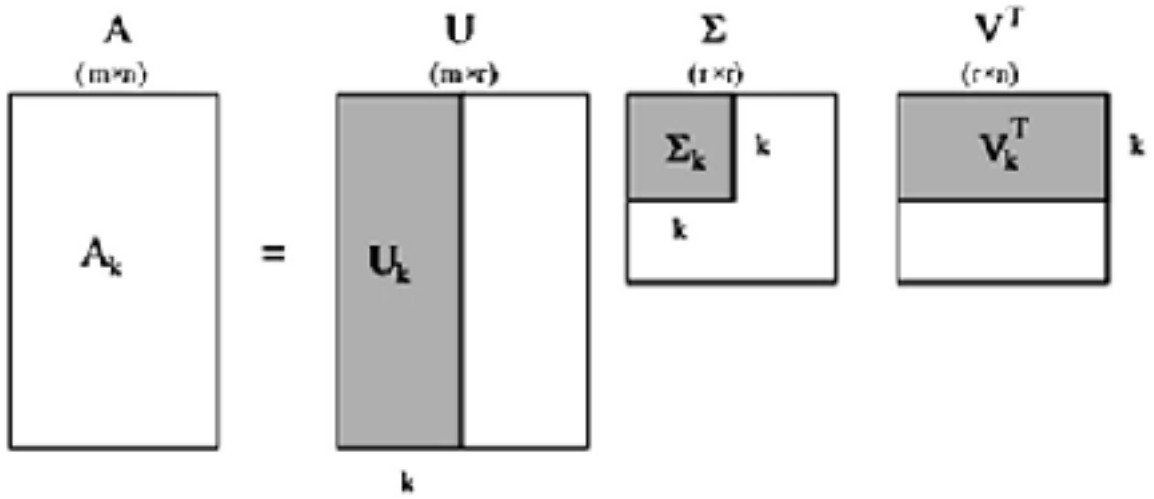
"*First, the SVD decomposes **A** into orthogonal factors that represent both types and documents. Vector representations for both types and documents are achieved simultaneously. Second, the SVD sufficiently captures the underlying semantic structure of a collection and allows for adjusting the representation of types and documents in the vector space by choosing the number of dimensions. Finally, the computation of the SVD is manageable for large datasets.*" **[6]**

How the SVD for a *m* × *n* type-by-document input matrix **A** with the rank (" *number of vectors in the basis of the column space or the vector subspace spanned by the column vectors* " **[6]**) of **A** = *r is decomposed* is defined as follows:

$$\mathbf{A} = \mathbf{U}\Sigma\mathbf{V}^{\mathrm{T}}$$

In which **U** is an orthogonal matrix, **V** is an orthogonal matrix, and Σis a diagonal matrix (Σ = diagonal(s_1, s_2, ..., s_n) with the remaining matrix cells all zeros **[7]**

A pictorial representation of the SVD of input matrix **A** and the best rank-k approximation to **A** can be seen in below **[8]**.



A can be written as the sum of rank 1 matrices **[9]**:

$$A = \sum_{i=1}^{r} u_i \sigma_i v_i^T .$$

r can be reduced to *k* to create

$$A_k = \sum_{i=1}^{k} u_i \sigma_i v_i^T$$

The matrix $\mathbf{A}_k$ is the closest rank *k* approximation to the original matrix **A [9] [10]**.

“ T*he matrix* $\boldsymbol{A}_k$ *(*$\boldsymbol{A}_k = \boldsymbol{U}_k S_k \boldsymbol{V}_k^T$*) is created by ignoring or setting equal to zero all but the first k elements or columns of the type vectors in* ***U****, the first k singular values in* S*, and the first k elements or columns of the document vectors in* ***V****. The first k columns of* ***U*** *and* ***V*** *are orthogonal, but the rows of* ***U*** *and* ***V****, the type and document vectors, consisting of k elements are not orthogonal. By reducing the dimension from r to k, extraneous information and variability in type usage, referred to as “noise,” which is associated with the database or document collection is removed. ”* **[6]**

$\mathbf{A}_k$ captures important underlying semantic structure of types and documents. Types similar in meaning are adjacent in *k*-dimensional vector space even though they never occur in a same part of a document, and documents similar in conceptual meaning are near each other even if they have similar types **[11]**. This *k*-dimensional vector space is the foundation for the semantic structures LSA operation.

In the previously mentioned example ,**[6]**

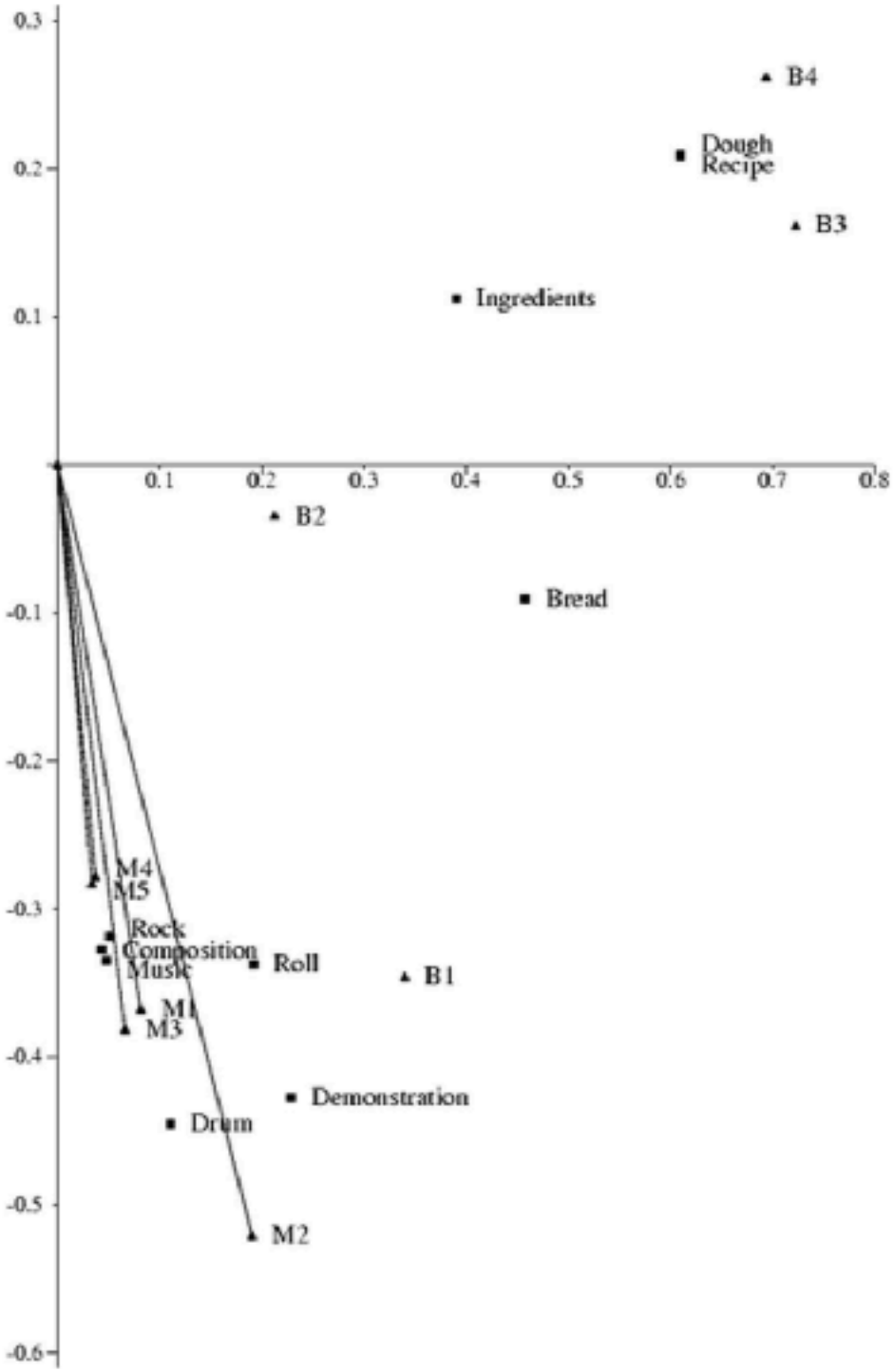
“ *Using the small document collection from Table 1 and its corresponding type-by-document matrix in Tables 2 and 3, the SVD can be computed and truncated to a two-dimensional vector space by reducing the rank to k = 2. Table 4 shows the SVD of the example type-by-document matrix given in Table 3. The values in the boldface cells in the type matrix* ***U****, the document matrix* ***V****, and the diagonal matrix of singular values* Σ *are used to encode the representations of types and documents in the two-dimensional vector space.*”

TABLE 4

Matrix U-Type Vectors									
Bread	**.42**	**−.09**	−.20	.33	−.48	−.33	.46	−.21	−.28
Composition	**.04**	**−.34**	.09	−.67	−.28	−.43	.02	−.06	.40
Demonstration	**.21**	**−.44**	−.42	.29	.09	−.02	−.60	−.29	.21
Dough	**.55**	**.22**	.10	−.11	−.12	.23	−.15	.15	.11
Drum	**.10**	**−.46**	−.29	−.41	.11	.55	.26	−.02	−.37
Ingredients	**.35**	**.12**	.13	−.17	.72	−.35	.10	−.37	−.17
Music	**.04**	**−.35**	.54	.03	−.12	−.16	−.41	.18	−.58
Recipe	**.55**	**.22**	.10	−.11	−.12	.23	−.15	.15	.11
Rock	**.05**	**−.33**	.60	.29	.02	.33	.28	−.35	.37
Roll	**.17**	**−.35**	−.05	.24	.33	−.19	.25	.73	.22
Matrix Σ-Singular Values									
	1.10	0	0	0	0	0	0	0	0
	0	**.96**	0	0	0	0	0	0	0
	0	0	**.86**	0	0	0	0	0	0
	0	0	0	**.76**	0	0	0	0	0
	0	0	0	0	**.66**	0	0	0	0
	0	0	0	0	0	**.47**	0	0	0
	0	0	0	0	0	0	**.27**	0	0
	0	0	0	0	0	0	0	**.17**	0
	0	0	0	0	0	0	0	0	**.07**
	0	0	0	0	0	0	0	0	0
Matrix V-Document Vectors									
M1	**.07**	**−.38**	.53	.27	.08	.12	.20	.50	.42
M2	**.17**	**−.54**	−.41	.00	.28	.43	−.34	.22	−.28
M3	**.06**	**−.40**	−.11	−.67	−.12	.12	.49	−.23	.23
M4	**.03**	**−.29**	.55	.19	−.05	.22	−.04	−.62	−.37
M5	**.03**	**−.29**	.27	−.40	−.27	−.55	−.48	.21	−.17
B1	**.31**	**−.36**	−.36	.46	−.15	−.45	.00	−.32	.31
B2	**.19**	**−.04**	.06	−.02	.65	−.45	.41	.07	−.40
B3	**.66**	**.17**	.00	.06	−.51	.12	.27	.25	−.35
B4	**.63**	**.27**	.18	−.24	.35	.10	−.35	−.20	.37

“ *Table 5 shows a rank-2, k = 2, plot of the types, represented by squares, and documents, represented by triangles, in the music and baking titles collection. Each point represents a type or document vector, a line starting at the origin and ending at a defined type or document point. The (x, y) pair is defined by x = first dimension or column of matrix* ***U*** *or* ***V*** *multiplied by the first singular value and y = second dimension or column of matrix* ***U*** *or* ***V*** *multiplied by the second singular value for type and document points, respectively. Looking at the vectors for the types and documents, the types most similar to each other and the documents most similar to each other are determined by the angles between vectors. If two vectors are similar, then they will have a small angle between them. In table 5, the documents M4, “A Perspective of Rock Mu- sic in the 90’s,” and M1 “Rock and Roll Music in the 1960’s” are the closest documents to document M3, “Drum and Bass Composition,” and yet they share no types in common. Similarly, the type vector for “music” is closest to type vectors “rock” and “composition,” however, the next closest type vector corresponds to the type vector for “drum.” This similarity is notable because “music” and “drum” never co-occur in the same document.* ”**[6]**

TABLE 5


0.3
0.2
0.1
-0.1
-0.2
-0.3
-0.4
-0.5
-0.6
0.1
0.2
0.3
0.4
0.5
0.6
0.7
0.8
B4
Dough Recipe
B3
Ingredients
B2
Bread
M4
M5
Rock
Composition
Music
Roll
B1
M1
M3
Demonstration
Drum
M2

Acknowledgement

Although the SVD is an easy way to explore the underlying meaning of texts, it will deviate from “ the actual recomputation of the reduced rank space using the same data to some extent ” because the update is based on $\mathbf{A}_k$ and not the original matrix **A. [12]**

Summary

Latent semantic analysis (LSA) uses reduced rank space model to exploit the latent semantic structure of type-document associations. Creating, calculating, and using the reduced rank vector space model is based on SVD .

References

[1] Firth, J.R. (1957). "A synopsis of linguistic theory 1930-1955". *Studies in Linguistic Analysis*. Oxford: Philological Society: 1–32. Reprinted in F.R. Palmer, ed. (1968). *Selected Papers of J.R. Firth 1952-1959*. London: Longman.

[2] Latent Semantic Analysis (Tutorial) - University of Victoria

[3] Witter, D., & Berry, M. W. (1998). Downdating the latent semantic indexing model for conceptual information retrieval. *The Computer Journal*, *41*, 589–601.

[4] Berry, M. W., & Browne, M. (2005). Understanding search engines: Mathematical model- ing and text retrieval (2nd ed.). Philadelphia: SIAM.

[5] Salton, G., & Buckley, C. (1991). Automatic text structuring and retrieval—experiments in automatic encyclopedia searching. *Proceedings of the 14th Annual International ACM SIGIR Conference on Research and Development in Information* Retrieval, pp. 21–30.

[6] Google Books: Handbook of Latent Semantic Analysis. July 22, 2017

[7] Golub, G., & Van Loan, C. F. (1989). *Matrix computations* (2nd ed.). Baltimore: Johns Hopkins Unversity Press.

[8] Berry, M. W., & Fierro, R. (1996). Low-rank orthogonal decompositions for information retrieval applications. *Numerical Linear Algebra With Applications*, *3*, 301–327.

[9] Björck, Å. (1996). *Numerical methods for least squares problems*. Unpublished manuscript, Linköping University, Linköping, Sweden.

[10] Berry, M. W., & Browne, M. (2005). Understanding search engines: Mathematical model- ing and text retrieval (2nd ed.). Philadelphia: SIAM.

[11] Berry, M. W., Dumais, S., & O'Brien, G. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, *37*, 573–595.

[12] Berry, M. W., Dumais, S., & O'Brien, G. (1995). Using linear algebra for intelligent information retrieval. *SIAM Review*, *37*, 573–595.