Latent Semantic Indexing (LSI)
A Fast Track Tutorial

Dr. Edel Garcia
admin@miislita.com

First Published on September 21, 2006; Last Update: October 21, 2006

Abstract

This fast track tutorial provides instructions for scoring queries and documents and for ranking results using a Singular Value Decomposition (SVD) calculator and the Term Count Model. The tutorial should be used as a quick reference for our SVD and LSI Tutorial series described at the following link:
http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-1-understanding.html

Keywords

latent semantic indexing, LSI, singular value decomposition, SVD, eigenvectors, documents, queries, cosine similarity, term count model

Background: The following LSI example is taken from page 71 of Grossman and Frieder's

Information Retrieval, Algorithms and Heuristics (1)
http://www.miislita.com/book-reviews/book-reviews.html

A "collection" consists of the following "documents"

d1: *Shipment of gold damaged in a fire.*
d2: *Delivery of silver arrived in a silver truck.*
d3: *Shipment of gold arrived in a truck.*

The authors used the Term Count Model to score term weights and query weights, so local weights are defined as word occurences. The following document indexing rules were also used:

- stop words were not ignored
- text was tokenized and lowercased
- no stemming was used
- terms were sorted alphabetically

In this tutorial we want to use this example to illustrate how LSI works. These days we know that most current LSI models are not based on mere local weights, but on models that incorporate local, global and document normalization weights. Others incorporate entropy weights and link weights. We also know that modern models ignore stop words and terms that occur once in documents. Term stemming and sorting in alphabetical order is optional. For this fast track tutorial, this example is good enough.

Problem: Use Latent Semantic Indexing (LSI) to rank these documents for the query *gold silver truck*.

Step 1: Score term weights and construct the term-document matrix **A** and query matrix:

Terms	d1	d2	d3	q
a	1	1	1	0
arrived	0	1	1	0
damaged	1	0	0	0
delivery	0	1	0	0
fire	1	0	0	0
gold	1	0	1	1
in	1	1	1	0
of	1	1	1	0
shipment	1	0	1	0
silver	0	2	0	1
truck	0	1	1	1

Step 2: Decompose matrix **A** matrix and find the **U**, **S** and **V** matrices, where

A = USV^T

For this example you may try a software like the Bluebit Matrix Calculator http://www.bluebit.gr/matrix-calculator/ (2), the JavaScript SVD Calculator http://users.pandora.be/paul.larmuseau/SVD.htm (3) or a software package like MathLab http://www.mathworks.com/ (4) or Scilab http://www.scilab.org/ (5). Note that these come with their own learning curves and sign conventions (*** See footnote**). Enter **A** in your preferred tool. For instance, from Bluebit output we can see that

U =

-0.4201	0.0748	-0.0460
-0.2995	-0.2001	0.4078
-0.1206	0.2749	-0.4538
-0.1576	-0.3046	-0.2006
-0.1206	0.2749	-0.4538
-0.2626	0.3794	0.1547
-0.4201	0.0748	-0.0460
-0.4201	0.0748	-0.0460
-0.2626	0.3794	0.1547
-0.3151	-0.6093	-0.4013
-0.2995	-0.2001	0.4078

S =

4.0989	0.0000	0.0000
0.0000	2.3616	0.0000
0.0000	0.0000	1.2737

V =

-0.4945	0.6492	-0.5780
-0.6458	-0.7194	-0.2556
-0.5817	0.2469	0.7750

V^T =

-0.4945	-0.6458	-0.5817
0.6492	-0.7194	0.2469
-0.5780	-0.2556	0.7750

Step 3: Implement a Rank 2 Approximation by keeping the first columns of **U** and **V** and the first columns and rows of **S**.

$$U \approx U_k = \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \qquad k = 2$$

$$S \approx S_k = \begin{bmatrix} 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix}$$

$$V \approx V_k = \begin{bmatrix} -0.4945 & 0.6492 \\ -0.6458 & -0.7194 \\ -0.5817 & 0.2469 \end{bmatrix} \qquad V^T \approx V_k^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \end{bmatrix}$$

Step 4: Find the new document vector coordinates in this reduced 2-dimensional space.

Rows of **V** holds eigenvector values. These are the coordinates of individual document vectors, hence

d1(-0.4945, 0.6492)
d2(-0.6458, -0.7194)
d3(-0.5817, 0.2469)

Step 5: Find the new query vector coordinates in the reduced 2-dimensional space.

$$q = q^T U_k S_k^{-1}$$

Note: These are the new coordinate of the query vector in two dimensions. Note how this matrix is now different from the original query matrix **q** given in **Step 1**.

$$q = q^T U_k S_k^{-1}$$

$$q = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} \frac{1}{4.0989} & 0.0000 \\ 0.0000 & \frac{1}{2.3616} \end{bmatrix} \qquad k = 2$$

$$q = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

Step 6: Rank documents in decreasing order of query-document cosine similarities.

The formula for computing cosine similarity values is given in The Classic Vector Space Model in http://www.miislita.com/term-vector/term-vector-3.html (6). Essentially we compute dot products between query and document vector coordinates and divide by the product of query and document vector lengths.

$$\mathbf{sim(q,\ d) = \frac{q \bullet d}{|q|\,|d|}}$$

$$\mathbf{sim(q,\ d_1)} = \frac{(-0.2140)\,(-0.4945) + (-0.1821)\,(0.6492)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.4945)^2 + (0.6492)^2}} = -0.0541$$

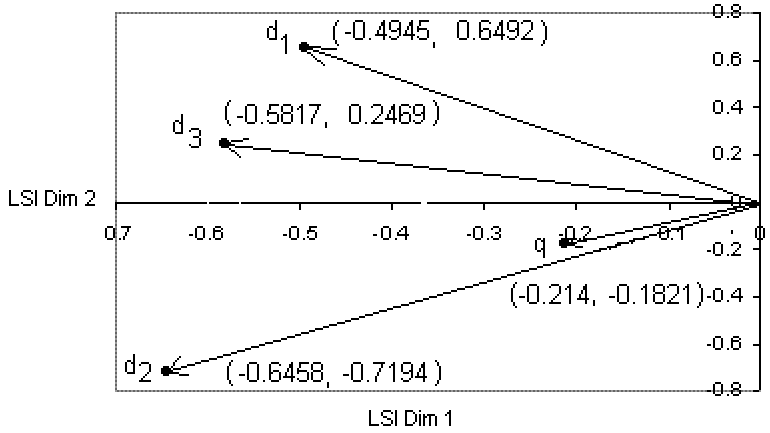
$$\mathbf{sim(q,\ d_2)} = \frac{(-0.2140)\,(-0.6458) + (-0.1821)\,(-0.7194)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.6458)^2 + (-0.7194)^2}} = 0.9910$$

$$\mathbf{sim(q,\ d_3)} = \frac{(-0.2140)\,(-0.5817) + (-0.1821)\,(0.2469)}{\sqrt{(-0.2140)^2 + (-0.1821)^2}\ \sqrt{(-0.5817)^2 + (0.2469)^2}} = 0.4478$$

Ranking documents in descending order

$$\mathbf{d_2 > d_3 > d_1}$$

We can see that document d2 scores higher than d3 and d1. Its vector is closer to the query vector than the other vectors. Also note that Term Vector Theory is still used at the beginning and at the end of LSI.



*** Please see BlueBit Important Upgrade**

Questions

1. Rework the example given in this tutorial, but this time remove all stopwords before constructing **A**. Define term weights as follows

   query: $w_{iq} = tf_{iq}$
   documents: $w_{ij} = tf_{ij}$

2. Rework the example given in this tutorial, but this time remove all stopwords before constructing **A**, Define term weights as follows

   query: $w_{iq} = tf_{iq}$
   documents: $w_{ij} = tf_{ij}*log(D/d_i)$

   where d_i is the number of documents containing term i. D is the collection size; in this case $D = 3$.
3. Repeat exercise 2. Define term weights as follows

   query: $w_{iq} = tf_{iq}$
   documents: $w_{ij} = tf_{ij}*log((D-d_i)/d_i)$

This fast track is aimed at readers of our SVD and LSI tutorial series. (7 – 10). For details or full explanation of concepts please refer to this series. You might also want to check our Singular Value Decomposition (SVD) Fast Track Tutorial (11), available at

http://www.miislita.com/information-retrieval-tutorial/singular-value-decomposition-fast-track-tutorial.pdf

* BlueBit Important Upgrade

Note 1 After this tutorial was written, BlueBit upgraded the SVD calculator and now is giving the $\mathbf{V}^T$ transpose matrix. We became aware of this today 10/21/06. This BlueBit upgrade doesn't change the calculations, anyway. Just remember that if using $\mathbf{V}^T$ and want to go back to **V** just switch rows for columns.

Note 2 BlueBit also uses now a different subroutine and a different sign convention, which flips the coordinates of the figures given above. Absolutely none of these changes affect the final calculations and main findings of the example given in this tutorial. Why? Read why here:

http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-4-lsi-how-to-calculations.html

References

1. Information Retrieval, Algorithms and Heuristics
   http://www.miislita.com/book-reviews/book-reviews.html
2. Bluebit Matrix Calculator
   http://www.bluebit.gr/matrix-calculator/
3. JavaScript SVD Calculator
   http://users.pandora.be/paul.larmuseau/SVD.htm
4. MathLab
   http://www.mathworks.com/
5. Scilab
   http://www.scilab.org/
6. The Classic Vector Space Model
   http://www.miislita.com/term-vector/term-vector-3.html
7. SVD and LSI Tutorial 4
   http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-4-lsi-how-to-calculations.html
8. SVD and LSI Tutorial 3
   http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-3-full-svd.html
9. SVD and LSI Tutorial 2
   http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-2-computing-singular-values.html
10. SVD and LSI Tutorial 1
    http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-1-understanding.html
11. Singular Value Decomposition (SVD) Fast Track Tutorial
    http://www.miislita.com/information-retrieval-tutorial/singular-value-decomposition-fast-track-tutorial.pdf