

LSI Keyword Research A Fast Track Tutorial

Dr. Edel Garcia
admin@miislita.com

First Published on October 18, 2006; Last Update: October 21, 2006
Copyright © Dr. E. Garcia, 2006. All Rights Reserved.

Abstract

This fast track tutorial provides instructions for conducting keyword research using co-occurrence theory, 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, term-term matrix, co-occurrence theory, seo myths

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.*

This is the same example we used in our previous fast track tutorial and described in

Latent Semantic Indexing (LSI) Fast Track Tutorial (2)
<http://www.miislita.com/information-retrieval-tutorial/latent-semantic-indexing-fast-track-tutorial.pdf>

In this tutorial we use the same experimental conditions (i.e., the Term Count Model), assumptions and limitations. We want to use this example to illustrate how LSI finds combination of terms by grouping these in a reduced space. A detailed explanation is described in

SVD and LSI Tutorial 5: LSI Keyword Research and Co-Occurrence Theory (3)
<http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-5-lsi-keyword-research-co-occurrence.html>

Problem: Use Latent Semantic Indexing (LSI) to cluster terms. Find also terms that could be used to expand or reformulate the query. Assume that the query is *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/> (4), the JavaScript SVD Calculator <http://users.pandora.be/paul.larmuseau/SVD.htm> (5) or a software package like MathLab <http://www.mathworks.com/> (6) or Scilab <http://www.scilab.org/> (7). Note that these come with their own learning curves and sign convention (* **See footnote**). Enter **A** in your preferred tool. For instance, from Bluebit output we can see that

$$\mathbf{U} = \begin{bmatrix} -0.4201 & 0.0748 & -0.0460 \\ -0.2995 & -0.2001 & 0.4078 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.1576 & -0.3046 & -0.2006 \\ -0.1206 & 0.2749 & -0.4538 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.4201 & 0.0748 & -0.0460 \\ -0.2626 & 0.3794 & 0.1547 \\ -0.3151 & -0.6093 & -0.4013 \\ -0.2995 & -0.2001 & 0.4078 \end{bmatrix} \quad \mathbf{S} = \begin{bmatrix} 4.0989 & 0.0000 & 0.0000 \\ 0.0000 & 2.3616 & 0.0000 \\ 0.0000 & 0.0000 & 1.2737 \end{bmatrix}$$

$$\mathbf{V} = \begin{bmatrix} -0.4945 & 0.6492 & -0.5780 \\ -0.6458 & -0.7194 & -0.2556 \\ -0.5817 & 0.2469 & 0.7750 \end{bmatrix} \quad \mathbf{V}^T = \begin{bmatrix} -0.4945 & -0.6458 & -0.5817 \\ 0.6492 & -0.7194 & 0.2469 \\ -0.5780 & -0.2556 & 0.7750 \end{bmatrix}$$

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

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

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

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

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

Rows of **U** holds eigenvector values. These are the coordinates of individual term vectors. Thus, from the reduced matrix (**U_k**)

Terms	Term Vector Coordinates	
a	-0.4201	0.0748
arrived	-0.2995	-0.2001
damaged	-0.1206	0.2749
delivery	-0.1576	-0.3046
fire	-0.1206	0.2749
gold	-0.2626	0.3794
in	-0.4201	0.0748
of	-0.4201	0.0748
shipment	-0.2626	0.3794
silver	-0.3151	-0.6093
truck	-0.2995	-0.2001

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

Using $\mathbf{q} = \mathbf{q}^T \mathbf{U}_k \mathbf{S}_k^{-1}$

$$\mathbf{q} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 & 1 \end{bmatrix} \begin{bmatrix} -0.4201 & 0.0748 \\ -0.2995 & -0.2001 \\ -0.1206 & 0.2749 \\ -0.1576 & -0.3046 \\ -0.1206 & 0.2749 \\ -0.2626 & 0.3794 \\ -0.4201 & 0.0748 \\ -0.4201 & 0.0748 \\ -0.2626 & 0.3794 \\ -0.3151 & -0.6093 \\ -0.2995 & -0.2001 \end{bmatrix} \begin{bmatrix} 1 & 0.0000 \\ 4.0989 & 0.0000 \\ 0.0000 & 2.3616 \end{bmatrix} = \begin{bmatrix} -0.2140 & -0.1821 \end{bmatrix}$$

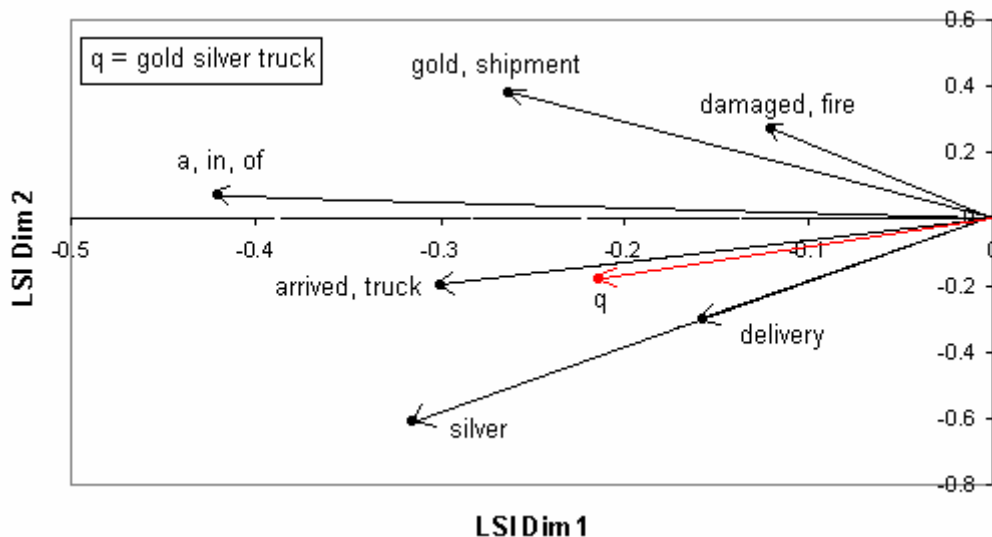
Step 6: Group terms into clusters.

Normally grouping is done by comparing cosine angles between any two pair of vectors. The formula for computing cosine similarities is given in

The Classic Vector Space Model (8, 9)
<http://www.miislita.com/term-vector/term-vector-3.html>

Since in this example we are dealing with a two-dimensional space, we can plot vectors and conduct a visual inspection. Obviously for more than three dimensions a visual representation is not possible and you would need to compute cosine similarities and sort these in descending order. This must be done for each reference vector.

Plotting vector coordinates,



*** Please See BlueBit Important Upgrade**

the following clusters are obtained:

- 1. *a, in of*
- 2. *gold, shipment*
- 3. *damaged, fire*
- 4. *arrived, truck*
- 5. *silver*
- 6. *delivery*

Some vectors are not shown since these are completely superimposed. This is the case of points 1 – 4. If unit vectors are used and small deviation ignored, clusters 3 and 4 and clusters 4 and 5 can be merged.

Step 7. Find terms that could be used to expand or reformulate the query.

The query is *gold silver truck*. Note that in relation to the query, clusters 1, 2 and 3 are far away from the query. Similarity wise these could be viewed as belonging to a “long tail”. If we insist in combining these with the query, possible expanded queries could be

<i>gold silver truck shipment</i>	<i>gold silver truck damaged</i>	
<i>gold silver truck shipment damaged</i>	<i>gold silver truck damaged in a fire</i>	
<i>shipment of gold silver truck damaged in a fire</i>		etc...

Looking around the query, the closer clusters are 4, 5, and 6. We could use these clusters to expand or reformulate the query. For example, the following are some of the expanded queries one could test.

<i>gold silver truck arrived</i>	<i>delivery gold silver truck</i>	
<i>gold silver truck delivery</i>	<i>gold silver truck delivery arrived</i>	etc...

Documents containing these terms should be more relevant to the initial query.

Questions

1. Do a search in a search engine in OR mode consisting in a two term query. Collect the top 5 titles. Consider these as documents. Construct an LSI term-document matrix. Use SVD to extract clusters of terms. Expand the query and resubmit this to the same search engine. Extract new clusters of terms.
2. Repeat exercise 1, but this time submitting the same queries in FINDALL mode. Explain any difference in the observed clusters. How does the query mode influence your results?

* 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 \mathbf{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.miiisita.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. Latent Semantic Indexing (LSI)Fast Track Tutorial
<http://www.miislita.com/information-retrieval-tutorial/latent-semantic-indexing-fast-track-tutorial.pdf>
3. SVD and LSI Tutorial 5: LSI Keyword Research and Co-Occurrence
<http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-5-lsi-keyword-research-co-occurrence.html>
4. Bluebit Matrix Calculator
<http://www.bluebit.gr/matrix-calculator/>
5. JavaScript SVD Calculator
<http://users.pandora.be/paul.larmuseau/SVD.htm>
6. MathLab
<http://www.mathworks.com/>
7. Scilab
<http://www.scilab.org/>
8. The Classic Vector Space Model
<http://www.miislita.com/term-vector/term-vector-3.html>
9. SVD and LSI Tutorial 1
<http://www.miislita.com/information-retrieval-tutorial/svd-lsi-tutorial-1-understanding.html>