The Stata Journal (*yyyy*) *vv*, Number *ii*, pp. 1–13

lsemantica: A Stata Command for Text Similarity based on Latent Semantic Analysis

Carlo Schwarz
University of Warwick
Coventry, United Kingdom
c.r.schwarz@warwick.ac.uk

Abstract. The `lsemantica` command, presented in this paper, implements Latent Semantic Analysis in Stata. Latent Semantic Analysis is a machine learning algorithm for word and text similarity comparison. Latent Semantic Analysis uses Truncated Singular Value Decomposition to derive the hidden semantic relationships between words and texts. `lsemantica` provides a simple command for Latent Semantic Analysis in Stata as well as complementary commands for text similarity comparison.

Keywords: st0001, lsemantica, machine learning, Latent Semantic Analysis, Latent Semantic Indexing, Truncated Singular Value Decomposition, text analysis, text similarity

1 Introduction

The semantic similarity of two text documents is a highly useful measure. Knowing that two documents have a similar content or meaning can, for example, help to identify documents by the same author, measure the spread of information or detect plagiarism. There are two main problems when attempting to identify documents with similar meaning. First, the same word can have different meanings in different context. Secondly, different words can have the same meaning in other contexts. As a result, using just the words counts in documents as a measure of similarity is unreliable. The alternative of reading and hand coding documents already becomes prohibitive for a few hundred documents.

Latent Semantic Analysis (LSA) provides a solution to this problem, by providing a reliable measure of semantic similarity for words and texts. LSA was developed by Deerwester et al. (1990) for the task of automated information retrieval in search queries. In search queries, it is important to understand the relationships and meanings of words, since just using the query terms often leads to unsatisfying search results. LSA improves search results, by taking into account the relationships and potential multiple meanings of words. It is this property that makes LSA applicable for a variety of different tasks among others:

1. Similarity of Words (Landauer et al. 1998)
2. Similarity of Texts (H.Gomaa and A. Fahmy 2013; Iaria et al. 2017)

 st0001

3. Computer Guided Summary Writing (Wolfe et al. 1998; Franzke et al. 2005)

4. Automated Essay Grading (Kintsch 2002; Miller 2003)

5. Assessing Discourse Coherence (Foltz 2007)

For all these applications LSA derives the hidden semantic connection between words and documents by using Truncated Singular Value Decomposition, the same transformation used in Principal Component Analysis. Principal Component Analysis uses Truncated Singular Value Decomposition to derive the components that explain the largest amount of variation in the data. In a similar vein, Truncated Singular Value Decomposition makes it possible for LSA to "learn" the relationships between words, by decomposing the "semantic space". This process allows LSA to accurately judge the meaning of texts. While LSA makes use of word co-occurrences, LSA can infer much deeper relations between words.

Landauer (2007) compares the workings of LSA to a problem of simultaneous equations. For two equations $A + 2B = 8$ and $A + B = 5$, neither equation alone is enough to infer the values of A or B. By combining the two equations together, it becomes straightforward to calculate the respective values for A and B. Similarly, the meaning of a document is based on the sum of the meaning of it's words: $meaning\_document = \sum(meaning_{word1}, \cdots, meaning_{word_n})$. For a computer, also in this context, it is not possible to infer the meaning of the words based on one document alone, but LSA is able to "learn" the meaning of words using the large set of simultaneous equations provided by all documents in the corpus.

As an illustration of the capabilities of LSA, Landauer (2007) provides the following example. On the one hand the text passages "A circle's diameter" and "radius of spheres" are judged by LSA to have similar meaning, despite having no word in common, on the other hand the text passage "music of the spheres" is judged as dissimilar by LSA, despite using similar words. As a result, text similarity comparison using LSA is preferable to just using the raw word counts in each document, since word-frequencies completely ignore multiple meanings of words. Furthermore, LSA also outperforms more recent machine learning algorithm, when it comes to document similarity comparison (Stevens et al. 2012).

This paper introduces the `lsemantica` command. `lsemantica` provides a solution for using LSA in Stata. `lsemantica` further facilitates the text-similarity comparison in Stata with the `lsemantica_cosine` command. In this way, `lsemantica` further improves the text analysis capabilities of Stata. Stata already allows to calculate the Levenshtein edit distance with the `strdist` command (Barker 2012) and the `txttool` command (Williams and Williams 2014) facilitates the cleaning and tokenizing of text data. Moreover, Schwarz (2017) and the `ldagibbs` command makes it possible to run Latent Dirichlet Allocation in Stata. While `ldagibbs` allows to group documents together by similar topics, `lsemantica` is preferable in cases where one is predominately interested in how similar documents are.

2 Decomposing the Semantic Space using Latent Semantic Analysis

This section describes how Truncated Singular Value Decompositions allows LSA to retrieve connections between words. As the first step, `lsemantica` creates a so called bag-of-words representation of the text data. In this process `lsemantica` creates a document-term-matrix A. The matrix A contains a row for each document $d \in D$ and a column for each unique term, i.e. words, in the vocabulary V. Each cell in A contains $f_{d,v}$ the number of times term v appears in document d:

$$\underbrace{A}_{D\times V} = \begin{pmatrix} f_{1,1} & \cdots & f_{1,d} \\ \vdots & \ddots & \vdots \\ f_{d,1} & \cdots & f_{d,v} \end{pmatrix}$$

The second step of `lsemantica` is to reweigh the word frequencies $f_{d,v}$ in the matrix A by their term-frequency-inverse-document-frequency (tf-idf). In this step $f_{d,v}$ is replaced by $tfidf(f_{d,v}) = (1 + \log(f_{d,v}))\cdot\left(\log\left(\frac{1+D}{1+d_v}\right) + 1\right)$, where d_v is the number of documents in which term v appears in. The tf-idf reweighting reduces the weights of words that appear in many documents, since these words are usually less important for the overall meaning of documents. After the tf-idf reweighting the matrix A contains:

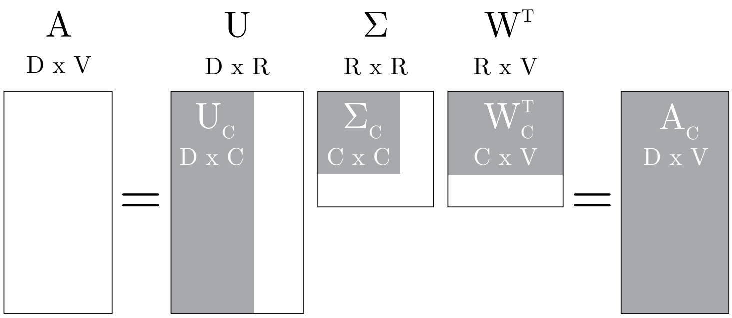
$$\underbrace{A}_{D\times V} = \begin{pmatrix} tfidf(f_{1,1}) & \cdots & tfidf(f_{1,d}) \\ \vdots & \ddots & \vdots \\ tfidf(f_{d,1}) & \cdots & tfidf(f_{d,v}) \end{pmatrix}$$

As the final step, `lsemantica` applies Singular Value Decomposition to the reweighted matrix A. Singular Value Decomposition transforms A, of rank R, into three matrices such that $A = U\Sigma W^T$, where U is a $D \times R$ orthogonal matrix, W^T is a $R \times V$ orthogonal matrix, and Σ is a $R \times R$ diagonal matrix. Afterwards, `lsemantica` truncates the resulting matrices by removing the rows and columns associated with the smallest eigenvalues in the matrix Σ. This truncation process reduces the dimensions of the matrices to a user-chosen number of components C, such that U becomes U_C of dimension $D \times C$, Σ becomes Σ_C of dimension $C \times C$, and W^T becomes W_C^T of dimension $C \times V$. The truncation process is represented in Figure 1.

The number of components C is usually chosen based on the size of the vocabulary. Martin and Berry (2007) suggest $100 \leq C \leq 1000$. During this truncation process, the entries of the original matrix A change as the number of components is reduced. In the end, this process results in the best rank-C approximation of the original matrix A called A_C. The truncation process of `lsemantica` is of utmost importance, since it reduces the components of the semantic space to the C most important ones.

The output of `lsemantica` is then based on $U_C \cdot \Sigma_C$, a $D \times C$ document-component matrix that can be used to compare the similarity of documents. The individual components of the document-component matrix represent the reduced dimensions of the

Figure 1: LSA Graphical Representation



Notes: This figure is a modified version of a figure in Martin and Berry (2007).

semantic space. These components capture the semantic relationships between the individual documents. Moreover, `lsemantica` can save the matrix W_C^T that allows to compare the similarity of words. As illustrated by the example in the introduction, even documents that contain completely different words can be judged to be similar by LSA, if the words appear together in similar semantic contexts.

3 Stata Implementation

This section describes how the `lsematica` command allows running LSA in Stata. To use `lsematica`, each document the user wants to use for LSA should be one observation in the data set. If documents consist of thousands of words, users can split the documents into smaller parts, e.g. paragraphs, to process each part separately. In any case, the text strings should be contained in one variable.[1] Furthermore, non-alphanumerical characters should be removed from the text strings.

The `lsemantica` command is started by specifying the variable that contains the text strings. There is an option to chose the number of components for Truncated Singular Value Decomposition. The options of `lsemantica` also allow to remove stopwords and short words from the text string. The `txttool` command (Williams and Williams 2014) provides more advanced text cleaning options, if required. `lsemantica` also provides two option to reduce the size of the vocabulary. These options are helpful in cases where Truncated Singular Value Decomposition requires a large amount of time due to the size of the vocabulary.

1. Since Stata 13, strL variables allow to store text strings up to 2-billion characters in one cell of the data set. Hence, there is hardly any limit on the amount of text that can be stored in one variable.

3.1 Syntax

`lsemantica` *varname* [, `components(`*integer*`)` `tfidf` `min_char(`*integer*`)` `stopwords(`*string*`)` `min_freq(`*integer*`)` `max_freq(`*real*`)` `name_new_var(`*string*`)` `mat_save` `path(`*string*`)`]

3.2 Options

`components(`*integer*`)` specifies the number of components the semantic space should be reduced to by `lsemantica`. The number of components is usually chosen based on the size of the vocabulary. The default is `components(`*300*`)`.

`tfidf` specifies if term-frequency-inverse-document-frequency reweighting should be used before applying the Truncated Singular Value Decomposition. In most cases tf-idf reweighting will improve the results.

Text Cleaning Options

`min_char(`*integer*`)` allows the removal of short words from the texts. Words with less characters than `min_char(`*integer*`)` will be excluded from LSA. The default is `min_char(0)`.

`stopwords(`*string*`)` specifies a list of words to exclude from `lsemantica`. Usually highly frequent words such as "I", "you", etc. are removed from the text, since these words contribute little to the meaning of documents. Predefined stopword lists for different languages are available online.

`min_freq(`*integer*`)` allows the removal of words that appear in few documents. Words that appear in fewer documents than `min_freq(`*integer*`)` will be excluded from LSA. The default is `min_freq(`*0*`)`.

`max_freq(`*real*`)` allows the removal of words that appear very frequently in documents. Words that appear in a share of more than `max_freq(`*real*`)` documents will be excluded from LSA. The default is `max_freq(`*1*`)`.

Output Options

`name_new_var(`*string*`)` specifies the name of the output variable created by `lsemantica`. These variables contain the topic assignments for each document. The user should ensure that `name_new_var(`*string*`)` is unique in the data set. If nothing is specified, the name of the variable is `name_new_var("component_")`, such that the names of the new variables will be component_1-component_C, where the C is the number of the components.

`mat_save` specifies if the word-component matrix should be saved. This matrix describes semantic relationships between words. By default, the matrix will not be saved.

`path(`*string*`)` sets the path where the word-component matrix is saved.

3.3 Output

`lsemantica` generates C new variables. These variables are the components generated by the Truncated Singular Value Decomposition. As described in the previous sections, these components capture the semantic relationships of documents and allow to calculate the similarity between documents.

The similarity of documents based on LSA is usually measured by the cosine similarity of the component vectors of each document. The cosine similarity of two documents $d1$ and $d2$ and their respective document-component vectors δ_{d1} and δ_{d2} is defined as:

$$cosine\_sim_{(d1,d2)} = \frac{\sum_{c=1}^{C} (\delta_{d1,c} \cdot \delta_{d2,c})}{\sqrt{\sum_{c=1}^{C} \delta_{d1,c}} \cdot \sqrt{\sum_{c=1}^{C} \delta_{d2,c}}}$$

The cosine similarity is hence the un-centered version of the correlation coefficient. The cosine similarity is 1 for perfect similarity documents and -1 for completely dissimilar documents. When using LSA, the cosine similarity usually lies within the unit interval. Only for highly dissimilar documents the cosine similarity will be negative.

`lsemantica` further provides the `lsemantica_cosine` command to facilitate the analysis of the cosine similarity. `lsemantica_cosine` calculates the cosine similarity for all documents in the corpus and stores it in Mata.[2] Furthermore, `lsemantica_cosine` can provide summary statistics for the cosine similarity and find highly similar documents. A separate help file explains the syntax of `lsemantica_cosine`.

4 Example

The example data set contains the title of 41,349 papers published in economic journals in the years 1980 until 2016. After loading the data, non-alphanumerical characters are removed from the title strings in preparation for LSA.

```
. use "$path/Data/example_data.dta", clear

.
. *********************************************
. *** 2) Run LSA and analyse Output
. *********************************************
. * combine title and summary
. gen text_strings =  title

.
.
. *remove non alpha numerical characters
. replace text_strings=strlower(text_strings)
(41,347 real changes made)
```

2. The cosine similarity matrix is stored in Mata since it is likely that the dimensions of the Matrix exceed the limits of Stata.

```
. replace text_strings = subinstr( text_strings, "." , " ", .)
(936 real changes made)

. replace text_strings = subinstr( text_strings, "!" , " ", .)
(32 real changes made)

. replace text_strings = subinstr( text_strings, "?" , " ", .)
(4,407 real changes made)

. replace text_strings = subinstr( text_strings, ":" , " ", .)
(12,193 real changes made)

. replace text_strings = subinstr( text_strings, ";" , " ", .)
(18 real changes made)

. replace text_strings = subinstr( text_strings, "," , " ", .)
(5,434 real changes made)

. replace text_strings = subinstr( text_strings, "(" , " ", .)
(291 real changes made)

. replace text_strings = subinstr( text_strings, ")" , " ", .)
(291 real changes made)

. replace text_strings = subinstr( text_strings, "&" , " ", .)
(211 real changes made)

. replace text_strings = subinstr( text_strings, `"""' , " ", .)
(274 real changes made)

. replace text_strings = subinstr( text_strings, "  " , " ", .)
(18,125 real changes made)

.
. replace text_strings = stritrim(text_strings)
(281 real changes made)

.
. list text_strings if _n<=10

          +----------------------------------------------------------------------+
          |                                                          text_strings |
          |----------------------------------------------------------------------|
       1. |                      what is labor supply and do taxes affect it     |
       2. |                      tax rules and the mismanagment of monetary policy |
       3. |   a consistent characterization of a near-century of price behavior  |
       4. | comparison of interwar and postwar business cycles monetarism rec..   |
       5. |                             trade policy as an input to development   |
          |----------------------------------------------------------------------|
```

Latent Semantic Analysis is then started by simply calling the `lsemantica` command. As the first step `lsemantica` prepares the documents and produces the document-term-matrix. During this preparation process `lsemantica` also removes words shorter 4 characters, words that appear in less than 10 documents or more than half of all documents from the data. Furthermore, stopwords are removed from the data. The resulting document-term-matrix is then reweighed using tf-idf. The command reports every time when 10% of the vocabulary have been processed.

```
. global stopwords "a able about across after all almost also am among an and any are as a
> t be because been but by can cannot could dear did do does either else ever every for fr
> om get got had has have he her hers him his how however i if in into is it its just leas
> t let like likely may me might most must my neither no nor not of off often on only or o
> ther our own rather said say says she should since so some than that the their them then
>  there these they this tis to too twas us wants was we were what when where which while
> who whom why will with would yet you your"
```

```
.
. lsemantica text_strings, components(300) min_char(4) min_freq(10) max_freq(0.5) tfidf  s
> topwords("$stopwords") mat_save path("$path")
*************************************
***** Latent Semantic Analysis ******
*************************************
Number of Components: 300
Minimal Word Length: 4
Minimal Word Frequency: 10
Maximal Word Frequency: .5
*************************************

  *************************************
  ******** Preparing Documents ********
  *************************************

  *** Creating Document-Word-Matrix ***
  Processing Vocabulary:
  10% done
  Processing Vocabulary:
  20% done
  (output omitted)
  Processing Vocabulary:
  90% done
  Processing Vocabulary:
  100% done
```

If documents do not have any words left after the text cleaning `lsemantica` will remove these observations from the data since they interfere with the Truncated Singular Value Decomposition. `lsemantica` reports which documents have been removed from the data as well as the size of the vocabulary. In the example, 167 documents are removed. The removal of documents is mainly due to the option `min_freq(`*10*`)`. The data set, for example, contains a paper simply titled “Notches”. Since the word “Notches” only appears in the title of 3 papers, it is not included in the vocabulary.

The next step of `lsemantica` is the calculation of the Truncated Singular Value Decomposition. Truncated Singular Value Decomposition is computationally intensive and can take some time. In some cases, Stata becomes unresponsive during this process. The time required for the Truncated Singular Value Decomposition increases with the size of the document-term-matrix and hence with the number of documents and the size of the vocabulary.

```
  The following observation where removed from the data, since they did not have any remai
> ning words:
               1
        +---------+
     1  |    240  |
     2  |    683  |
  (output omitted)
   167  |  41324  |
        +---------+

  Size of Vocabulary:
  3029
```

```
Now moving to SVD: this may take a while!

*************************************
******* Running Truncated SVD *******
*************************************
```

After lsemantica is finished running, one can begin to analyze the similarity of documents by calculating the cosine similarity between the component vectors using the lsemantica_cosine command.[3] The resulting cosine similarity matrix is only stored in Mata due to its dimensions. lsemantica_cosine allows to calculate the average similarity as well as the maximal and minimal similarity to other paper titles.

```
. lsemantica_cosine component_1-component_300 ,  mean_cosine min_cosine max_c
> osine find_similar(10) find_similar_cosine(10)
.
.
.
.
. sum mean_similarity, d
                       mean_similarity
-------------------------------------------------------------

      Percentiles      Smallest
 1%     .0044491       .0018937
 5%     .0064587       .0021306
10%     .0078202       .0021863       Obs              41,182
25%     .0106267       .0023644       Sum of Wgt.      41,182

50%     .0144946                      Mean           .0151279
                        Largest       Std. Dev.      .0060271
75%     .0189352       .0414024
90%     .0232734       .0423894       Variance       .0000363
95%     .0260268       .0429952       Skewness       .5782946
99%     .0315863       .0432296       Kurtosis       3.186935
.
.
. sum max_similarity, d
                       max_similarity
-------------------------------------------------------------

      Percentiles      Smallest
 1%     .5652832       .3455565
 5%     .6286974       .3464915
10%     .6642829       .3655767       Obs              41,182
25%     .7306858       .3674786       Sum of Wgt.      41,182

50%      .816731                      Mean           .8242296
                        Largest       Std. Dev.      .1226226
75%      .946547              1
90%     .9941671              1       Variance       .0150363
95%     .9985264              1       Skewness      -.1298808
99%            1              1       Kurtosis       2.128631
.
.
```

3. The lsemantica_cosine command is memory intensive, since the command multiplies the document-component matrix with itself.

```
. sum min_similarity, d
                        min_similarity
-------------------------------------------------------------

      Percentiles      Smallest
 1%    -.2512143      -.4081201
 5%    -.1909725      -.4081201
10%    -.1618612      -.4010201       Obs              41,182
25%    -.1235881      -.4010201       Sum of Wgt.      41,182

50%    -.0936972                      Mean           -.1047622
                        Largest       Std. Dev.       .0433816
75%    -.0743099      -.0296275
90%    -.0625703      -.0294599       Variance        .001882
95%    -.0569326      -.0284645       Skewness      -1.634287
99%    -.0480057      -.0281307       Kurtosis       7.055067
```

Furthermore, `lsemantica_cosine` can find the most similar papers for each of the papers in the data. In the example, the 10 most similar papers are calculated. Afterwards, the 5 most similar paper titles for the first paper in the data are listed. One can see that LSA accurately identified highly similar papers all discussing questions of labor supply.

```
. list most_similar_* cosine_most_similar_* if _n ==1

     +-----------------------------------------------------------------------+
  1. | most_s~1 | most_s~2 | most_s~3 | most_s~4 | most_s~5 | most_s~6 |
     |    28468 |      513 |     6288 |    27267 |    34267 |    26707 |
     |----------+----------+----------+----------+----------+----------|
     | most_s~7 | most_s~8 | most_s~9 | most_~10 | cosine_~1 | cosine_~2 |
     |    38850 |    29305 |    29322 |    38932 | .81473917 | .81221156 |
     |-----------+-----------+-----------+-----------+-----------|
     | cosine_~3 | cosine_~4 | cosine_~5 | cosine_~6 | cosine_~7 |
     | .67660235 | .67439987 | .67347484 | .66513775 | .66482448 |
     |-----------------------+-----------------------+-----------------------|
     |       cosine_~8       |       cosine_~9       |       cosine~10       |
     |       .66331657       |       .66331657       |       .66203841       |
     +-----------------------------------------------------------------------+

.
. list title if _n==1 | _n==513 | _n==28468 | _n==27267 | _n==6288 | _n==3426
> 7

       +---------------------------------------------------------------------+
       | title                                                               |
       |---------------------------------------------------------------------|
    1. | What Is Labor Supply and Do Taxes Affect It?                        |
  513. | Family Labor Supply with Taxes                                      |
 6288. | The Effect of Taxes on Labor Supply in the Underground Economy      |
27267. | Low-Skilled Immigration and the Labor Supply of Highly Skilled W..  |
28468. | Labor Supply and Taxes: A Survey                                    |
       |---------------------------------------------------------------------|
34267. | Worktime Regulations and Spousal Labor Supply                       |
       +---------------------------------------------------------------------+
```

`lsemantica` makes it possible to calculate the number of papers the original paper is highly similar to. In the example, a cut-off for the cosine similarity of 0.75 was chosen. The Mata code generates a new variable called high_sim_papers containing the number of papers that have a cosine similarity above this cut-off. The example data also contain the number of citation for each paper. Hence, one can estimate a regression of the number of similar papers on the number of citations. The regression reveals a significant positive relationship between the two variables.

```
. sort pub_year

.
. mata
------------------------------------------------- mata (type end to exit) ---
: pub_year = st_data(. , "pub_year")

:
: high_sim_paper= J(0,1,.)
: for (y=1980 ; y<=2016 ; y++){
>
>         cosine_submat = select(cosine_sim, pub_year:==y)
>         cosine_submat = select(cosine_submat´,pub_year:>=y )´
>
>         high_sim = rowsum(( cosine_submat:>=J(rows(cosine_submat) , cols(co
> sine_submat) , 0.75) ))
>
>         high_sim_paper = high_sim_paper \ high_sim
> }

:
: var = st_addvar("double", "high_sim_paper")
: st_store(. , "high_sim_paper" ,high_sim_paper)

:
: end
------------------------------------------------------------------------------

.
.
. reg citations high_sim_paper i.pub_year
```

Source	SS	df	MS
Model	34061173.2	37	920572.248
Residual	476018071	41,144	11569.5623
Total	510079244	41,181	12386.2763

Number of obs	=	41,182
F(37, 41144)	=	79.57
Prob > F	=	0.0000
R-squared	=	0.0668
Adj R-squared	=	0.0659
Root MSE	=	107.56

citations	Coef.	Std. Err.	t	P>|t|	[95% Conf.	Interval]
high_sim_paper	.0982082	.0456843	2.15	0.032	.008666	.1877504
pub_year						
1981	19.001	15.02938	1.26	0.206	-10.45691	48.4589
(output omitted)						
2016	-43.48352	10.7657	-4.04	0.000	-64.58452	-22.38252
_cons	47.07542	10.51478	4.48	0.000	26.46623	67.68461

Finally, `lsemantica` makes it possible to compare semantic relationships and the similarity of words. Using `lsemantica_word_comp` one can import the word-component matrix stored by `lsemantica`. Again `lsemantica_cosine` can be used to calculate the cosine between the words in the data and find the most similar words. The example shows that `lsemantica` identifies that the words 'labor', 'force', 'segmented', 'division', 'frictional' as well as 'monopsony' are related to each other.

```
. lsemantica_word_comp using "$path/word_comp.mata"
.
.
. lsemantica_cosine component_1-component_300,  find_similar(10) find_similar
> _cosine(10)
.
. * "labour":  _n ==1516
. list most_similar_* cosine_most_similar_* if _n ==1516
```

1516. most_s~1	most_s~2	most_s~3	most_s~4	most_s~5	most_s~6
1768	812	1144	2465	1111	291
most_s~7	most_s~8	most_s~9	most_~10	cosine_~1	cosine_~2
1394	1309	2253	2232	.53823032	.46088074
cosine_~3	cosine_~4	cosine_~5	cosine~6	cosine_~7	
.33327173	.30747563	.26425962	.2354069	.23372151	
cosine_~8	cosine~9	cosine~10			
.21986212	.2002355	.19779614			

```
.
. list word if _n==1516 | _n==812  | _n==1144 | _n==2465 | _n==1111 | _n==176
> 8
```

	words
812.	division
1111.	force
1144.	frictional
1516.	labor
1768.	monopsony
2465.	segmented

5 References

Barker, M. 2012. STRDIST: Stata module to calculate the Levenshtein distance, or edit distance, between strings. *Statistical Software Components* .

Deerwester, S., S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41(6): 391.

Foltz, P. W. 2007. Discourse Coherence and LSA. *Handbook of latent semantic analysis* 167–184.

Franzke, M., E. Kintsch, D. Caccamise, N. Johnson, and S. Dooley. 2005. Summary

Street®: Computer support for comprehension and writing. *Journal of Educational Computing Research* 33(1): 53–80.

H.Gomaa, W., and A. A. Fahmy. 2013. A Survey of Text Similarity Approaches. *International Journal of Computer Applications* 68(13): 13–18.

Iaria, A., C. Schwarz, and F. Waldinger. 2017. Frontier Knowledge and Scientific Production Evidence from the Collapse of International Science. *Forthcoming Quarterly Journal of Economics* .

Kintsch, W. 2002. The potential of latent semantic analysis for machine grading of clinical case summaries. *Journal of Biomedical Informatics* 35(1): 3–7.

Landauer, T. K. 2007. LSA as a theory of meaning. *Handbook of latent semantic analysis* 3–34.

Landauer, T. K., P. W. Foltz, and D. Laham. 1998. An Introduction to Latent Semantic Analysis. *Discourse Processes* 25(2-3): 259–284.

Martin, D. I., and M. W. Berry. 2007. Mathematical foundations behind latent semantic analysis. *Handbook of latent semantic analysis* 35–56.

Miller, T. 2003. Essay Assessment with Latent Semantic Analysis. *Educational Computing Research* 29(4): 495–512.

Schwarz, C. 2017. ldagibbs: A command for Topic Modeling in Stata using Latent Dirichlet Allocation. *Forthcoming Stata Journal* .

Stevens, K., P. Kegelmeyer, D. Andrzejewski, and D. Buttler. 2012. Exploring topic coherence over many models and many topics. *Proceedings of the 2012 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning* .

Williams, U., and S. P. Williams. 2014. txttool: Utilities for text analysis in Stata. *Stata Journal* 14(4): 817–829.

Wolfe, M. B. W., M. E. Schreiner, B. Rehder, D. Laham, P. W. Foltz, W. Kintsch, and T. K. Landauer. 1998. Learning from text: Matching readers and texts by Latent Semantic Analysis. *Discourse Processes* 25(2-3): 309–336.

About the authors

Carlo Schwarz is a PhD student at the University of Warwick. His research interests are in the field of applied microeconomics and political economy, with a focus on text analysis and machine learning. (www.carloschwarz.eu)