Information Retrieval

REPORT

Submitted by

Lakshya Sethi	Vineet Yadav	Ajay Jajoo
20	40	03

Contents

Analyzed tab:

Academic

Number of pages crawled:

174

Number of PDFs: *4*

Pages/PDFs skipped:

None

Introduction

A web-crawler has been created in python which crawls **Academic tab** to **one hop** present on the Delhi University's official website.

The crawler crawls through all the links present on the Academic webpage and downloads the source code of all the links into a text file on the local machine. Crawler also searches for the links of all the PDF files and if the content is not an image, crawler reads the PDF file and downloads it into the local machine which is then converted to a text file.

After crawling through the webpages, a corpus is created using all the text files with the help of **NLTK** library. Preprocessing is being done over the corpus, Term frequency matrix is calculated, and hence, a TF-IDF matrix is generated. Based on TF-IDF matrix cosine similarity between all the documents is generated and the two documents with maximum cosine similarity are highlighted.

With the resultant cosine similarity matrix, considering each document as an initial data point, Agglomerative clustering is being done and hence a dendrogram is generated.

At last, TD-IDF matrix is given as input **(C)** to compute the SVD.

$$C = U_{mxm}\ S_{mxn}\ V^{T}_{nxn}$$

Where m is the number of terms in the corpus, n is the number of documents in the corpus. Finally, $\mathbf{C_k}$, a k-ranked matrix is calculated.

$$C_k = S_k V^{T}_{k}$$

Software/Libraries used

- ## Requests

  Requests allows to send HTTP/1.1 requests using Python. With it, we can add content like headers, form data, multipart files, and parameters via simple Python libraries. It also allows us to access the response data of Python in the same way.

- ## Urllib/Urllib2

  The urllib module in Python 3 allows to access websites via our program. Urllib in Python 3 is slightly different than urllib2 in Python 2, but they are mostly the same. Through urllib, we can access websites, download data, parse data, modify your headers, and do any GET and POST requests we might need to do.

- ## Beautiful Soup

  Beautiful Soup is a Python library for pulling data out of HTML and XML files. It works with any of our favorite parser to provide idiomatic ways of navigating, searching, and modifying the parse tree.

- ## OS

  The OS module in Python provides a way of using operating system dependent functionality. The functions that the OS module provides allows you to interface with the underlying operating system that Python is running on – be that Windows, Mac or Linux.

- RE

  Regular expressions (called REs, or regexes, or regex patterns) are essentially a tiny, highly specialized programming language embedded inside Python and made available through the **re** module. Using this little language, you specify the rules for the set of possible strings that you want to match; this set might contain English sentences, or e-mail addresses, or TeX commands, or anything you like.

- Glob

  The **`glob`** module finds all the pathnames matching a specified pattern according to the rules used by the Unix shell, although results are returned in arbitrary order. It is more powerful than os.listdir that does not use wildcards.

- PyPDF2

  PyPDF2 is a pure-python PDF library capable of splitting, merging together, cropping, and transforming the pages of PDF files. It can also add custom data, viewing options, and passwords to PDF files. It can retrieve text and metadata from PDFs as well as merge entire files together.

- NLTK

  The NLTK module is a massive tool kit, aimed at helping us with the entire Natural Language Processing (NLP) methodology. NLTK aid us with everything from splitting sentences from paragraphs, splitting up words, recognizing the part of speech of those words, highlighting the main subjects, and then even with helping our machine to understand what the text is all about.

- Sklearn

  Scikit-learn is a free software machine learning library for the Python programming language. It features various classification, regression and clustering algorithms including support vector machines, random forests, gradient boosting, k-means and DBSCAN, and is designed to interoperate with the Python numerical and scientific libraries NumPy and SciPy. In this project, Sklearn is used to do Agglomerative Clustering of the documents based on cosine similarity matrix.

- SciPy

  SciPy is an open source Python library used for scientific computing and technical computing. SciPy contains modules for optimization, linear algebra, integration, interpolation, special functions, FFT, signal and image processing, ODE solvers and other tasks common in science and engineering. Here SciPy is used to compute cosine distances between the documents.

- Plotly

  Plotly is an online collaborative data analysis and graphing tool. The Python API allows you to access all Plotly's functionality from Python. Plotly figures are shared, tracked, and edited all online and the data is always accessible from the graph. Here, Plotly is used to display dendrogram and network structure.

- Matplotlib

  Matplotlib is a plotting library for the Python programming language. It provides an object-oriented API for embedding plots into applications using general-purpose GUI toolkits. Here, matplotlib is used to plot a scatter graph of the documents in K-dimensional space of reduced SVD.

Preprocessing

Software used for preprocessing

```
from bs4 import BeautifulSoup
from nltk.corpus import stopwords
```

- Extracting only text from the source code

```
source_code = requests.get(link)
plain_text = source_code.text
soup = BeautifulSoup(plain_text,'lxml')
for script in soup("script"):
        script.extract()

for css in soup("style"):
        css.extract()
text = soup.get_text(" ")
```

BeautifulSoup is used to extract only the text from the source code of a webpage. It discards scripts, html tags and css scripts to get mainly the content of the webpage.

- Other preprocessing steps

```
for filename in glob.glob(os.path.join(path, '*.txt')):
    fout = open(filename, 'rb')
    corpus_list.append(fout.read().decode('utf-8').lower())

processed_corpus = []
for sentence in corpus_list:
    stop_words = set(stopwords.words("english"))
    words  =  tokenizer.tokenize(sentence)
    processed_sentence =
        [i for i in words if not i in stop_words and not i.isnumeric() and len(i) > 1]
    processed_corpus.append(processed_sentence)
```

1. Stopwords are being removed from the corpus using inbuilt stopwords dictionary from nltk.corpus.
2. All the numbers are removed from the text.
3. Words with length greater than one are considered.
4. Case-folding: all the text is converted to lowercase.

- Size of Term Document Matrix

Number of terms: 6797

Number of documents: 178

∴ Size of the resultant TDM is 6797 x 178

Similarity Matrix in VSM

Cosine similarity matrix

```
for documents in enumerate(matrix_array):
    for doc in enumerate(matrix_array, start = documents[0]+1):
        result = 1 - spatial.distance.cosine(documents[1], doc[1])
        cosine_similarity_matrix.append(result)
    cosine_similarity_matrix_2D.append(cosine_similarity_matrix)
```

Cosine similarity matrix is calculated for all the pairs of documents i.e., 31, 506 values are generated and of which 15, 753 are unique values as the generated cosine similarity matrix is a symmetric matrix.

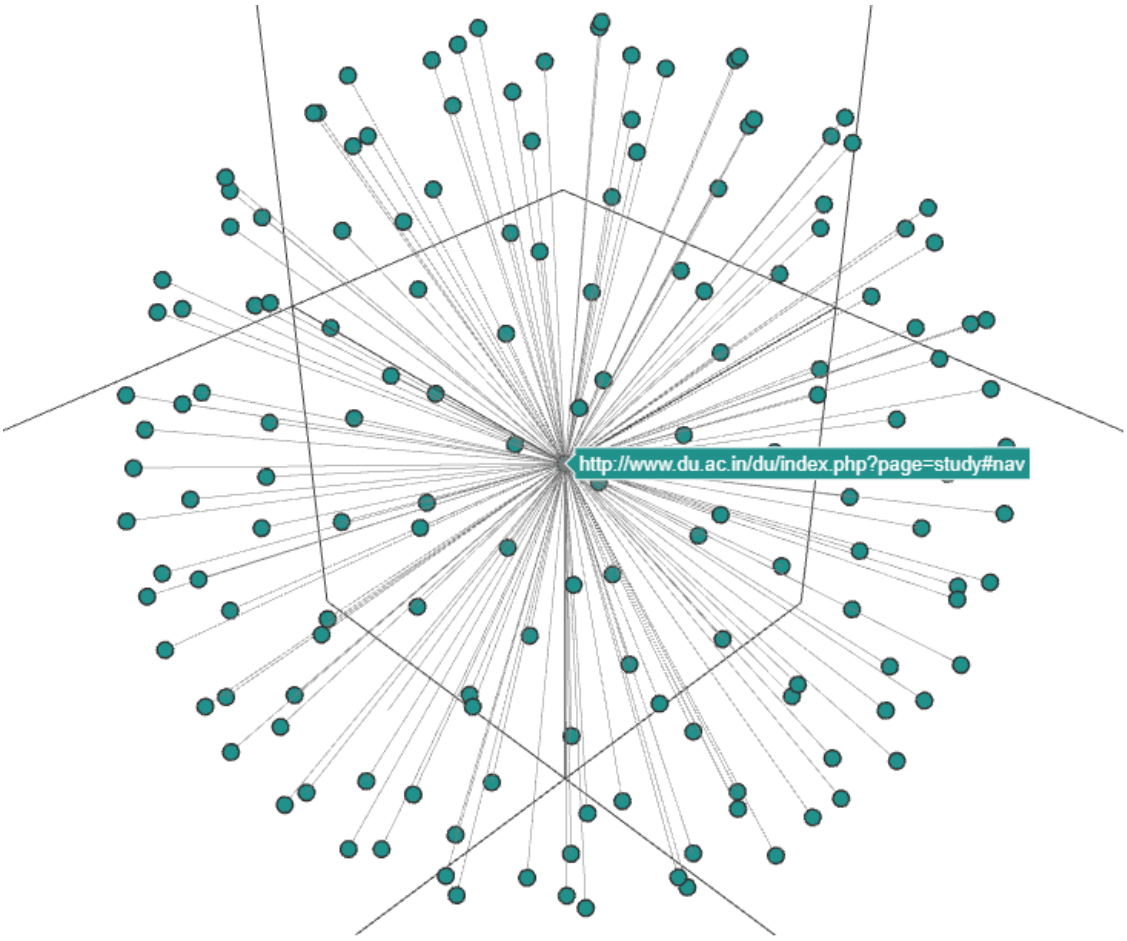
A webpage with name: Cosine Similarity Matrix.html contains the matrix with number at the top representing document with highest similarity. The row and column with highest similarity is highest in the webpage.

Documents with Highest similarity matrix are document number 114.txt and 120.txt with cosine similarity of 0.995939494485.

					114.txt			
	0.117656653173	0.719680778704	0.458735765618	0.577403112111	0.0555334530638	0.81353953864	0.789436507311	0.0661981552463
	0.244396925639	0.0831086957052	0.0619024166011	0.0636324492406	0.00993536500735	0.0867869558798	0.0924626459999	0.0616924897707
	0.0910386283965	0.51149226019	0.364827665704	0.40862133949	0.0395638582429	0.573472734611	0.543239784129	0.0387624995994
	0.105408949381	0.632016207762	0.387693976902	0.462989610483	0.0504567622193	0.642096245489	0.620837123676	0.0541213712398
	0.0	0.019389131663	0.00506931867124	0.00837881881122	0.000478800092685	0.0116117618708	0.0127810650539	0.0
	0.100827702639	0.676939293789	0.402588722205	0.502085301024	0.0664163355615	0.713913876225	0.692223720587	0.0583257386267
	0.0750697484896	0.456038131947	0.308390877315	0.35466096486	0.0467810507364	0.500077593467	0.495942472159	0.0379978572743
	0.125540082071	0.770485506508	0.490890257066	0.610105754068	0.0619626893909	0.870591542438	0.845646358974	0.0675505307287
	0.109809538575	0.679747715008	0.444141170078	0.532110911025	0.0577569411605	0.762361109025	0.738478011458	0.059516518162
	0.0989146914429	0.505208209798	0.321123813341	0.397731020297	0.0472758876179	0.569270513653	0.553310917183	0.0569137828374
	0.0750048244591	0.53866400811	0.349875049919	0.416045719322	0.0629441353189	0.6226490772	0.601465226347	0.0418111447783
	0.061556883303	0.343281420692	0.390312974908	0.229975828587	0.104357725762	0.330802626873	0.340745656753	0.0235313994444
	0.0828956299741	0.511480301489	0.340256908199	0.420376525208	0.037436610336	0.556879654216	0.544309536583	0.0445952679597
120.txt	0.0179738850286	0.0679037812908	0.0502134431309	0.0442046566857	0.995939494485	0.0515631618937	0.0555828457758	0.00370796051168
	0.121011731186	0.748551296493	0.470514148554	0.581928143821	0.0591276283233	0.832869692478	0.805936889774	0.0654044353506
	0.106564742221	0.629061358185	0.417054405877	0.500845541702	0.0455354461331	0.701144680953	0.684751485332	0.0545980852492

Network Structure

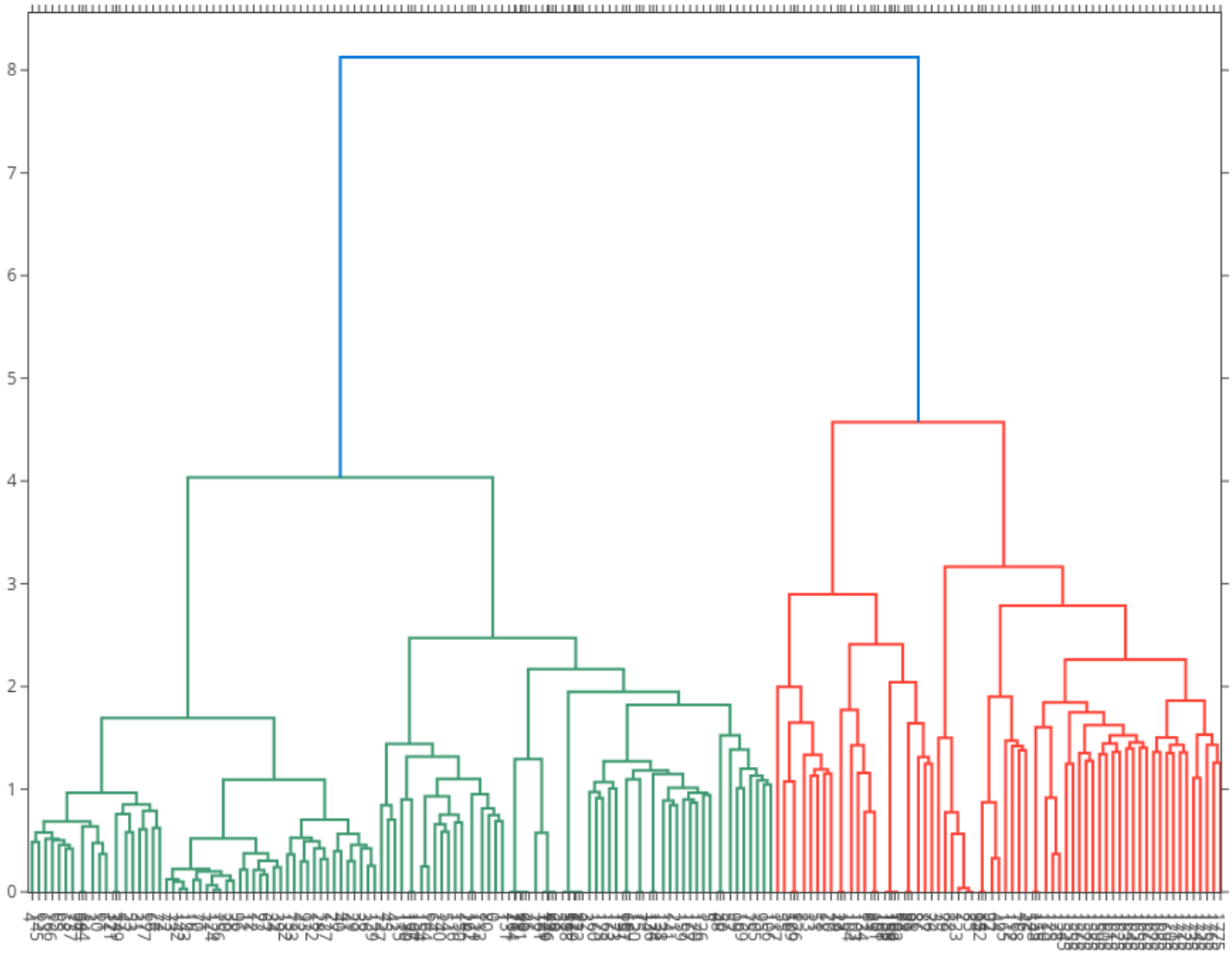
- Network structure till one hop



A network structure is created with root node as the first link i.e., http://www.du.ac.in/du/index.php?page=study and all other nodes are the links that are present on the Academic webpage.

Network structure is present inside the Crawler_level_1 folder.

Agglomerative Clustering-Dendrogram



Dendrogram of 178 documents are created in Plotly. Initially, each document is considered as a data point and based on most similar documents, documents are iteratively merged to form a dendrogram.

Analysis of 5 most similar documents

S. No.	Document One (With Links)	Document Two (With Links)	Cosine Similarity
1.	114.txt Tender Quotations	120.txt Tender Archive	0.995939494485
	The documents are very similar as Tender Archive is under one of the tabs of Tender Quotations.		
2.	162.txt Finance - DSSW	163.txt Finance – Hear them out!	0.991439781761
3.	113.txt Finance Website	162.txt Finance - DSSW	0.987178352343
4.	113.txt Finance Website	163.txt Finance – Hear them out!	0.986624457647
5.	165.txt Finance – Ragging Prevention	162.txt Finance - DSSW	0.985243123207
	The pair of documents in all 4 rows are almost identical because these webpages contain images but with different titles. Because images are discarded by crawler, minor difference in the contents of webpages is mainly because of different titles. Otherwise, all the links belong to the webpage of Finance.		

Singular Value Decomposition

TF-IDF Input matrix (C)

TF-IDF matrix is given as weighted input matrix. The dimensions of TF-IDF matrix is 6797x178.

Where 6797 are the number of terms and 178 are the number of documents in the corpus.

SVD Term Matrix (U)

SVD Term matrix (U) is calculated using scipy.linalg library module. Returned U matrix is of the form 6797x178.

```
U, s, Vh = SL.svd(tfidf.tfidf_matrix.toarray().transpose(), full_matrices=False)
```

```
[[ -7.66554663e-05  -1.15057919e-04   1.49420527e-04 ...,
9.78705959e-04
    8.89173884e-01   2.97607103e-02]
 [ -8.21060696e-05  -2.44746516e-04  -5.83193945e-04 ...,
3.35417584e-03
   -4.14293018e-01   9.90361369e-02]
 [ -9.69676800e-04  -4.42352173e-04   2.66830382e-03 ...,
1.18248883e-03
    1.34688891e-02  -3.89838617e-03]
 ...,
 [ -3.83277332e-05  -5.75289594e-05   7.47102636e-05 ...,
1.65474369e-04
   -3.03387745e-04  -3.82096015e-04]
 [ -2.52251569e-03  -5.18654742e-03  -1.30982424e-02 ...,
1.82237670e-02
   -1.70284422e-03  -1.86208281e-02]
 [ -3.21938900e-05  -1.27813832e-04   3.00095468e-04 ...,
4.94574218e-05
    2.42356677e-06  -1.66576627e-04]]
```

Singular Value Matrix (S)

Singular value matrix (S) is calculated using scipy.linalg library module. Returned S matrix is a diagonal matrix of dimensions 178x178 which is min(number of terms(M), number of documents(N)).

```
[  8.32702021e+00   2.37813720e+00   2.11950515e+00   2.00843079e+00
   1.72991714e+00   1.66685232e+00   1.62382984e+00   1.53368360e+00
   ...

   5.79307859e-16   5.79307859e-16   5.79307859e-16   5.79307859e-16
   5.12509535e-16   3.49263660e-16]
```

SVD Document Matrix (V^T)

SVD Document matrix (V^T) is calculated using scipy.linalg library module. Returned V^T matrix is of the form 6797x178.

```
[[ -8.15073769e-02  -8.15073769e-02  -1.35595638e-02 ...,  -
7.42100780e-02
   -3.77963579e-02  -9.45984354e-02]
 [ -1.96623491e-02  -1.96623491e-02  -3.79180997e-03 ...,
8.55687521e-03
   -1.90979323e-03   9.62779968e-03]
 [  2.34374598e-01   2.34374598e-01   1.21125989e-02 ...,  -
7.18981417e-03
    2.91862601e-03   2.34159780e-02]
 ...,
 [  0.00000000e+00   9.58861245e-02   6.04984812e-17 ...,
1.73472348e-16
    1.46334740e-02   2.77555756e-17]
 [ -8.65454649e-01   2.61840903e-01   0.00000000e+00 ...,
8.67361738e-18
    8.82174427e-04  -2.34187669e-17]
 [  0.00000000e+00   2.74922642e-02   7.93093889e-17 ...,
4.85722573e-17
   -2.30353444e-02  -1.52655666e-16]]
```

Retrieval of top 5 documents

In the S diagonal matrix, apart from top 5 rows all are zeroed out and hence by using the formula

$$C_5 = S_5 V^T_5$$

Reduced SVD matrix is generated of dimensions 5x178

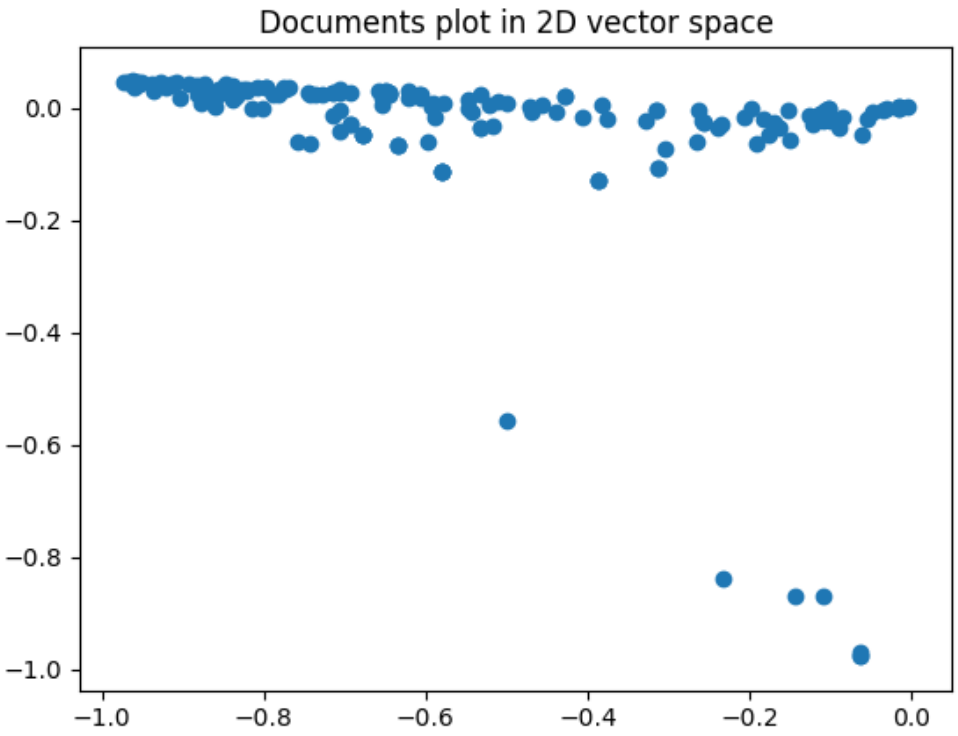
```
[[ -6.78713575e-01  -6.78713575e-01  -1.12910761e-01  -8.73819493e-01

   ...
   -8.83487038e-01  -6.21776322e-01  -5.22190437e-01  -6.17948820e-01
   -3.14731036e-01  -7.87723084e-01]


 [  2.72215589e-03   2.72215589e-03  -3.08689428e-02   2.38178040e-03
    3.87293951e-03  -5.22539481e-03  -3.66642629e-02  -1.30740958e-02
   ...
   -6.41330660e-02   6.77598826e-03]]
```

Scatter graph

Following graph is the scatter graph of SVD in 2-dimensional space i.e., using rank of two.



Analysis of 5 most similar documents

S. No.	Document One (With Links)	Document Two (With Links)	SVD
1.	114.txt Tender Quotations	120.txt Tender Archive	0.975593711147
2.	162.txt Finance - DSSW	163.txt Finance – Hear them out!	0.957726357521
3.	113.txt Finance Website	162.txt Finance - DSSW	0.9248144150315
4.	113.txt Finance Website	163.txt Finance – Hear them out!	0.9119928962023
5.	165.txt Finance – Ragging Prevention	162.txt Finance - DSSW	0.9012431232076
	∴ Top 5 documents according to both the SVD and Cosine Similarity matrix are same. It might be because the difference in the webpages are only based on titles and even the additional benefit of semantic analysis by SVD doesn't change the final result.		

Submitted by

Lakshya Sethi

20

lakshya.mcs16.du@gmail.com

Ajay Jajoo

03

ajay.mcs16.du@gmail.com

Vineet Yadav

40

vineet.mcs16.du@gmail.com