Clustering and Latent Semantic Indexing using Singular Value Decomposition on Reuters and MovieLens Datasets

February, 2015

Illinois Institute of Technology, Chicago

Prepared By: Aditya Kundu

1. **Working with the data**:

The data set we are working on in this experiment comes from the Reuter's collection of news articles. It is a subset of a larger Reuter's dataset of 21 thousand plus articles. Our data set has 91 total topics and a total of 10 thousand plus documents. Every topics has a certain number of documents ranging from 1 to 2877. The topics 'earn' and 'acq' are the largest topics in the data set with 2877 and 1650 number of documents. More information about the original data set can be found here: http://www.daviddlewis.com/resources/testcollections/reuters21578/readme.txt

There is also one interesting topic 'unknown' with 1830 number of documents. These are the documents that did not belong to a particular topic and were joined together as one 'unknown' topics. For our analysis using this topic will not be helpful because on basis of the terms used in these documents they may belong to a topic that we are using and can be misclassified.

Below is a list of some of the topics and the number documents that belong to each topic with the number of unique terms before and after removing stop words.

TOPIC	No. of Documents	No. of unique terms	No. of unique terms after removing stop words
Acq	1650	9060	8984
Bop	75	1566	1497
Carcass	50	1500	1432
Cocoa	55	1600	1529
Coffee	111	2359	2287
Corn	181	2589	2517
Cpi	69	1451	1383
Crude	389	4684	4609
Dlr	131	1972	1900
Earn	2877	9114	9041
Gnp	101	2293	2222
Gold	94	1688	1622
Grain	433	4137	4061
Interest	347	2956	2882
Livestock	75	1837	1769
Money-fx	538	4060	3985
Money-supply	138	1476	1412
Nat-gas	75	2054	1984
Oilseed	124	2240	2171
Reserves	55	1091	1029
Ship	197	3281	3208

Soybean	78	1883	1814
Sugar	126	2487	2415
Trade	369	4288	4213
Veg-oil	87	1778	1710
Wheat	212	2801	2727
All Topics combined	8637	20259	20181

From the above table we can see that the number of documents in a topic is proportional to the number of terms in those document i.e. higher the number of documents, higher the number of terms. However one interesting thing to be noted is that when we compute the total number of unique terms there is an overlap of mutual words. Many words commonly occur among different topics.

In our analysis we read all the 8637 documents to create a corpus 'alldata'. Preprocess this data set and make full document term matrices. For our later analysis we will subset this data in two parts. First set will be a set of topics with similar number of documents. It will include all documents from 'crude', 'grain' and 'trade' with 389, 433 and 369 number of documents respectively. We will call this data set **Data1** in further discussion. The second data set will be a set of topics with imbalanced number of documents. It will include all documents from topic 'acq' and 'wheat' with 1650 and 212 number of documents respectively. We will refer to this data set as **Data2** in further discussion.

2. Pre-processing

Preprocessing of text documents is a series of simple commands to remove the un-important values from the text that can be problematic in running our experiments. It includes the following processes:

```
alldata <- tm_map(alldata, tolower)          ## Convert to Lower Case

alldata <- tm_map(alldata, removeWords, stopwords("english"))  ## Remove
Stopwords

alldata <- tm_map(alldata, removePunctuation)      ## Remove Punctuations

alldata <- tm_map(alldata, stemDocument)                 ## Stemming

alldata <- tm_map(alldata, removeNumbers)                ## Remove Numbers

alldata <- tm_map(alldata, stripWhitespace) ## Eliminate Extra White
Spaces
```

2.1 Creating the document-term matrix

```
alldata <- DocumentTermMatrix(alldata)
```

A document term matrix shows the count of each term in a document with the rows representing documents and columns representing terms. Such document term matrices for large corpora like, in our data set, have high sparsity i.e. there are more empty values than non-empty values in the matrix. In our experiment the sparsity is 100%. We can lower the sparsity of the document term matrix by removing sparse terms in R as follows:

```
alldata <- removeSparseTerms(alldata, 0.9)
```

The measurement '0.9' means that terms that are not in more than 90% of the documents will be removed. The sparsity comes out to be 95% and the number of terms cuts down to 1123.
However using counts of terms per document may give us a matrix representation of the data they do not help us very well in finding relevant words. Many common words like articles, conjunctions, common verbs etc are bound to have higher frequency in a document than other words. Therefore using counts can be deceiving. This problem can be solved by using term frequency-inverse document frequency weight instead of counts.

It is a simple transformation which can be given by the given formula:

$$tf'_{ij} = tf_{ij} * \log \frac{m}{df_i},$$

where df_i is the number of documents in which the i^{th} term appears and is known as the **document frequency** of the term. This transformation is known as the **inverse document frequency** transformation.

In R we can do the above transformation as follows:

```
alldata <- weightTfIdf(alldata)
```

Using this transformation we reduce the weight of importance for words that appear more frequently in a document and increase it for words that unique to a document and relatively less frequent. Such words define a document better than others. We perform this transformation for Data1 and Data2 as well. After this step we have a document term matrix with tfidf as weights 'alldata'.

3. Experimentation

In this section we will cluster our document-term matrices using k-means and use these clustering results to compare with results after performing Latent Semantic Indexing using Singular Value Decomposition. The main idea is to map our documents and terms in semantic space to find certain hidden concepts. Before we compute SVD for our data we cluster the tfidf document term matrices for Data1 and Data2 using k-means with different number of clusters. This clustering result will be used to evaluate the performance of LSA on clustering after we perform SVD. Thus this clustering result will be our baseline to find out if LSA gave us better results or not.

3.1 Computing Latent Semantic Space using SVD

The function svd(M) in R takes an arbitrary matrix argument, M, and calculates the singular value decomposition of M. This consists of a matrix of orthonormal columns U with the same column space as M, a second matrix of orthonormal columns V whose column space is the row space of M and a diagonal matrix of positive entries D such that M = U * D * V'

In our experiment we perform SVD on the entire data set (8637 documents) instead of running it on subsets of the data to get better and larger concept space. The matrices U, D and V have special properties. When using a document term matrix where documents are rows and terms are columns, matrix U gives us importance of documents in the concept space, and V gives us importance of terms in concept space. Both the matrices map documents and terms in the latent sematic space respectively. The matrix D is a diagonal matrix of the eigen values.

SVD not only helps in bringing out the hidden concepts in the data but also helps in reducing dimensionality of the data. After computing SVD for the document term matrix we can reduce the dimensions of the data by reducing number of columns of U and V, and reducing both the number of columns and rows of D. While doing this we make sure that the reduced number of rows and columns are compatible in order to make the following matrix multiplication:

$$New.Mat = U^i \times D^i \times transpose(V^i)$$

Where i is the number of dimensions used.

The 'New.Mat' represents a near equivalent of the original matrix. Such a transformation works so well that we are able to retain the important information in the data, given we choose an appropriate number of dimensions/concepts.

To figure out a good number of dimensions to use, we experiment using different numbers and try to how many features give us a good measure of the data. But before we perform clustering we need to normalize our document vectors (rows of U).

Normalization is important in LSA since it is a variance maximizing exercise. It projects original data onto directions which maximize the variance. Since LSA seeks to maximize the variance of each component and the covariance may be large for a particular concept it will give a very high importance to that concept. Thus, for finding features usable for our analysis, an LSA without normalization would perform badly.

We normalize our document vectors by scaling the U matrix from our SVD in R as follows.

```
Scaled.U <- scale(svd$u)
```

Now our document vectors are normalized and we can perform clustering on our decomposed matrices.

3.2 K-means Clustering with different number of features

To estimate a good number of features to use we cluster our documents using different number of features and clusters. To evaluate for separate datasets we take the appropriate subset of rows (documents) from the matrix U that belong to the topics in our datasets 'Data1' and 'Data2'. In our data 'alldata' we know that the first 1650 documents belong to acq and last 212 belong to 'wheat'. We take the appropriate rows from U of svd and combine this to get this 'Data2' for clustering after SVD. In R this can performed as follows:

```
# subset of u for   crude(389 docs)   grain(433 docs)      trade(369 docs)

udata1 <- rbind( scaled.u[2191:2580,], scaled.u [5783:6216,],  scaled.u
[7970:8338,])

# subset of u for   acq(1650 docs)   wheat(212 docs)

udata2 <- rbind(   scaled.u [1:1650,],   scaled.u [8425:8637,])
```

After building these subsets for U we compute the near equivalent data matrix of various dimensions as per the formula discussed before for both Data1 and Data2. Then we cluster this matrix with K-Means. We test using different number of clusters and features. Below is an example In R with 50 features and 3 clusters.

```
test <- udata1[,1:50] %*% d[1:50,1:50] %*% t(svd$v[,1:50])

cl1 <- kmeans(test, 3)

plotcluster(test, cl1$cluster)
```

3.3 Motivation for Experiments

There are three key points to consider here.

1. LSA is supposed to provide a better representation and also better similarity measure. As discussed before that LSA provides a better representation of our data because it brings out the hidden concepts. Other than that it also gives us similarity measures for our documents and terms. It also gives us a concept space to which we can map our documents and terms in order to find their importance and relevance to a concept.

2. K-Means is based on similarities
   K-Means clustering clusters similar object together in Euclidean space. More similar the objects are the closer they are to each other. Therefore most similar objects belong to one cluster and dissimilar objects belong to separate clusters. It can also tell us which clusters are similar as they will be close to each other than those that are very dissimilar.

3. LSA should improve K-Means
   From our conceptual and theoretical understanding of LSA and K-Means we can say that clustering after LSA should perform better than clustering without LSA. From point 1 and 2 and our experiments we can understand why this is true.

Below are the clustering results of before and after LSA for both Data1 and Data2 with different number of clusters and features. We first start with clusters before LSA.

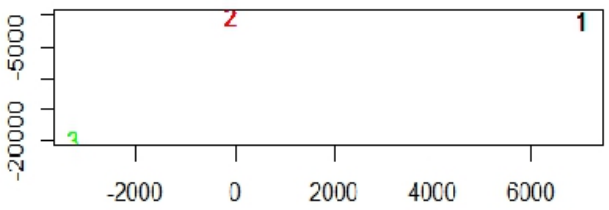

Fig 1. Clustering on Data1 with 3 clusters

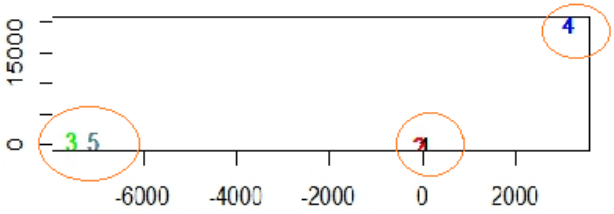

Fig 2. Clustering on Data1 with 5 clusters

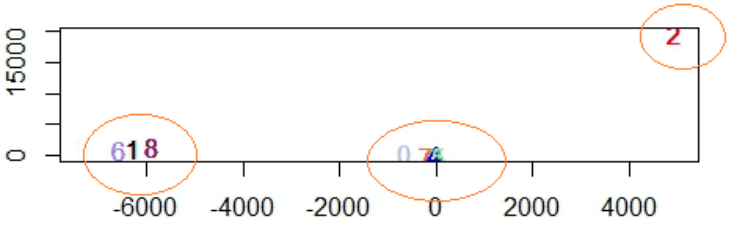

Fig 3. Clustering on Data1 with 10 clusters

From the above plots we can see that there is a set of 3 conceptual clusters in our data. This makes sense because there are 3 topics in the data set that we are working on. Even when the number of clusters for k-means is chosen as 5 or 10 we see that some documents, even though part of different clusters are still plotted close to others because of their similarity.

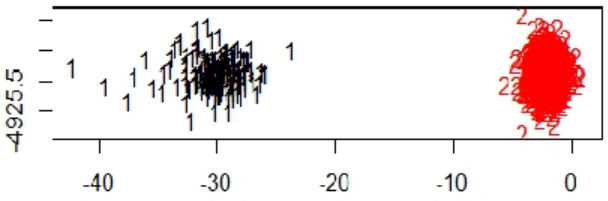

Fig 14. Cluster of Data2 with 2 clusters

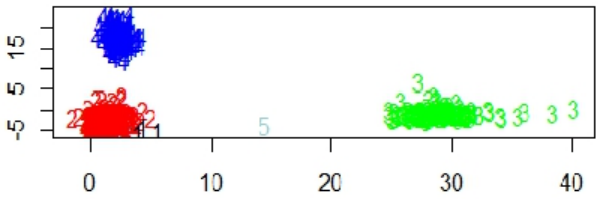

Fig 15. Cluster of Data2 with 5 clusters

Fig 15. Cluster of Data2 with 10 clusters

An interesting thing to notice here is that unlike the plots of Data1 here the logical number of clusters with different number of clusters is not the same. In the first plot we can see that there are two separate clusters corresponding to the two topics in the data set. However, in the second and the third plot we see additional logical clusters. Due to a large number of documents in one of the topic this may indicate that there are some documents that talk about more specific topics under 'acquisition'. Another interesting thing here is the top right cluster in Fig 15. This cluster has a significant number of clusters which have their centroids very close to each other.

Clusters after LSA for Data1

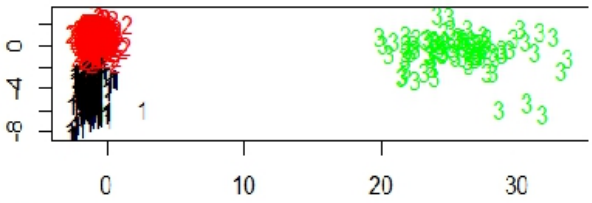

Fig 4. Clustering with 200 concepts and 3 clusters

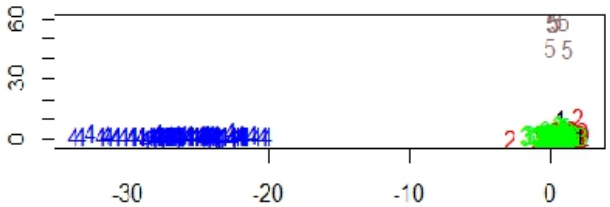

Fig 5. Clustering with 200 concepts and 5 clusters

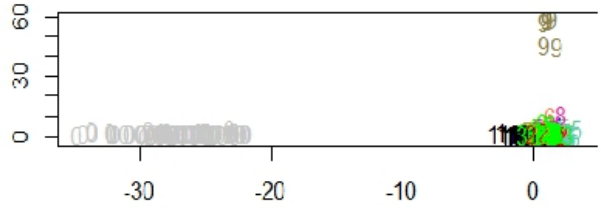

Fig 6. Clustering with 200 concepts and 10 clusters

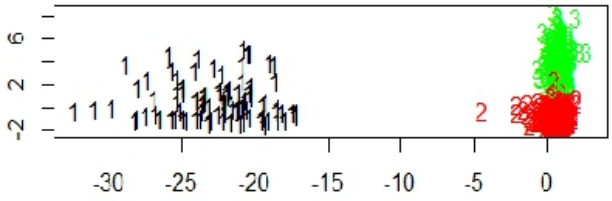

Fig 7. Clustering with 100 concepts and 3 clusters

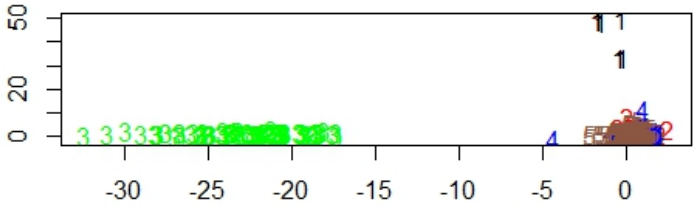

Fig 8. Clustering with 100 concepts and 5 clusters

Comparing figure 7 and 8 we see that a lot of data points from the top right cluster came down to the bottom right cluster leaving the top right cluster with very low number of data points. This may suggest an overlap in the terms that describe these topics and thus the shift in the data points. However this is an example of poor clustering.

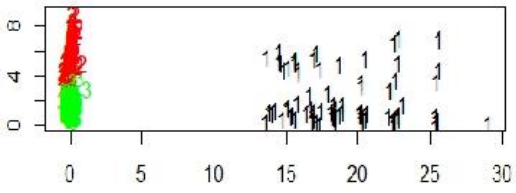

Fig 9. Clustering with 10 concepts and 3 clusters

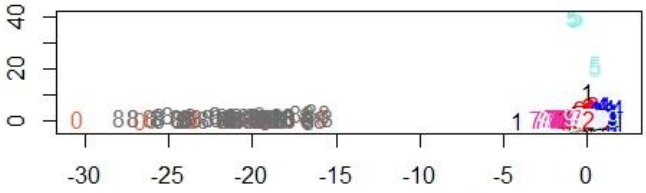

Fig 10. Clustering with 10 concepts and 10 clusters

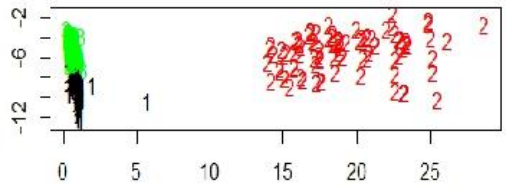

Fig 11. Clustering with 5 concepts and 3 clusters

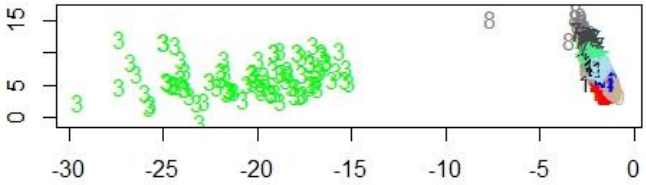

Fig 12. Clustering with 5 concepts and 10 clusters

Overall we can see that most of the clustering results were easily able to show three logical clusters indicating the three topics that we have chosen. However this does not mean that the clustering accuracy was good. In some cases the clusters had a significantly low number of data points whereas we know that the number of documents for each topic in our data set is comparable.

Clusters for Data2 after LSA

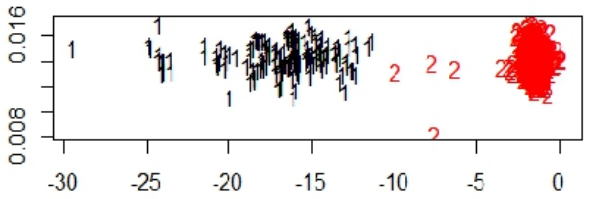

Fig 16. Clustering with 100 features and 2 clusters

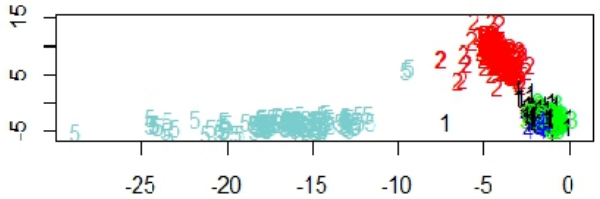

Fig 17. Clustering with 100 features and 5 clusters

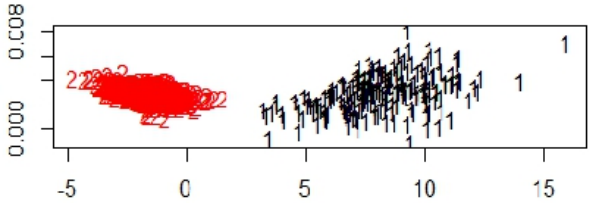

Fig 18. Clustering with 50 features and 2 clusters

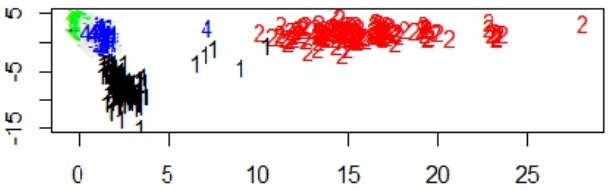

Fig 19. Clustering with 50 features and 5 clusters

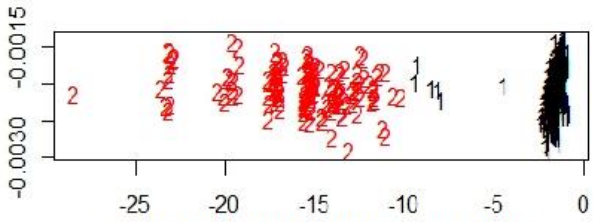

Fig 20. Clustering with 20 features for 2 clusters

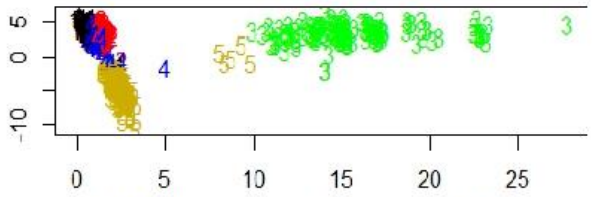

Fig 21. Clustering with 20 features for 5 clusters

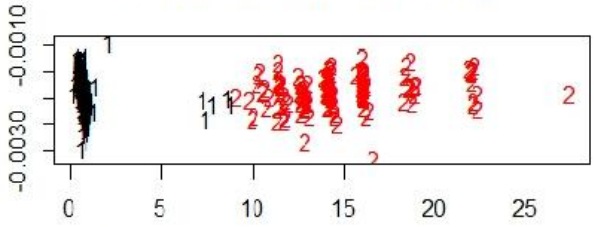

Fig 22. Clustering with 10 features for 2 clusters

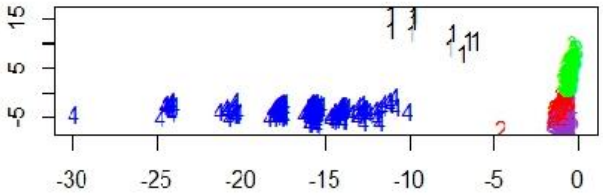

Fig 23. Clustering with 10 features for 5 clusters

4. Results and Evaluation

4.1 Evaluation Measures

For evaluation we will use two measures, Sum of Square Errors and Accuracy

Sum of Squared Errors, SSE is great to compare different clustering results for exactly the same data and number of clusters. However, we vary the number of clusters and the number of features. So, if we have more clusters, the SSE measure will be lower, just because of the way SSE is computed. Similarly, if we have more features, the SSE measure will be higher. Even the value of features matters. For example, SSE varies for document matrix using counts and using tf-idf just because of the difference in value magnitude if document vectors are not normalized.

A plot of the SSE against a series of sequential cluster levels can provide a useful graphical way to choose an appropriate cluster level. That is, an appropriate cluster solution could be defined as the solution at which the reduction in SSE slows dramatically. This produces an "elbow" in the plot of SSE against cluster solutions. In our example shown above, there is an "elbow" at the 3 cluster solution suggesting that solutions >3 do not have a substantial impact on the total SSE.

Accuracy

For the data sets that we used in our experiments, since we know the topic labels for the documents that we have chosen we can evaluate the performance of our clustering by calculating the accuracy. This means to calculate what percentage of documents have been correctly labeled by our clustering algorithm.

A simple way of calculating the accuracy for our data is to compare the labels we already know with the labels given by our clustering algorithm. We can table the class labels with the cluster vector given by the kmeans object and build a confusion matrix. Using the true positives and tru negatives in this confusion matrix we can calculate the accuracy as follows:

The confusion matrix:

		Predicted	
		Negative	Positive
Actual	Negative	**a**	**b**
	Positive	**c**	**d**

Then the accuracy AC can be calculated as:

$$AC = \frac{a+d}{a+b+c+d}$$

In our example of Data2 data set we know that the first 1650 documents belong to the topic 'acq' and the next 212 documents belong to 'wheat'. We create a new vector 'label' with first 1650 values as 1 and the next 212 as 2. Then using the following code we can create a confusion matrix.

```
label <- matrix(1,nrow=1862)

label[1651:1862] = 2

table(label, cl1$cluster)
```

The table function will print our confusion matrix as below:

```
> table(label, cl1$cluster)

label    1    2
    1 1500  150
    2  199   13
```

Now in this case the accuracy will be ((1500 + 13)/1862) * 100 = 81.256

This means that 81.265 percent of all the documents have been classified correctly.

Topics in the Corpus	No of clusters	No of concepts(k)	Sum of Squared Errors (SSE)	Accuracy per cluster over no. of clusters (%)
Crude + Grain + Trade	3	Before SVD	425.68	68.65
Crude + Grain + Trade	3	5	787.54	64.97
Crude + Grain + Trade	3	10	956.78	71.83
Crude + Grain + Trade	3	20	1124.36	73.58
Crude + Grain + Trade	3	50	1356.32	72.55
Crude + Grain + Trade	3	100	1533.12	71.18
Crude + Grain + Trade	5	Before SVD	486.82	65.32
Crude + Grain + Trade	5	5	881.63	62.11
Crude + Grain + Trade	5	10	1036.21	68.91
Crude + Grain + Trade	5	20	1388.98	68.91
Crude + Grain + Trade	5	50	1688.78	72.36
Crude + Grain + Trade	5	100	1723.33	69.52
Crude + Grain + Trade	10	Before SVD	560.67	63.13
Crude + Grain + Trade	10	5	954.74	55.64
Crude + Grain + Trade	10	10	1224.65	66.86
Crude + Grain + Trade	10	20	1532.23	69.25
Crude + Grain + Trade	10	50	1636.48	68.82
Crude + Grain + Trade	10	100	1818.71	63.59
Acq + wheat	2	Before SVD	14.88	76.32
Acq + wheat	2	5	47.02	69.90
Acq + wheat	2	10	93.53	69.90
Acq + wheat	2	20	113.14	76.93
Acq + wheat	2	50	151.02	75.09
Acq + wheat	2	100	201.312	75.87
Acq + wheat	3	Before SVD	25.75	74.22
Acq + wheat	3	5	47.75	63.87

Acq + wheat	3	10	82.34	69.85
Acq + wheat	3	20	149.55	71.18
Acq + wheat	3	50	182.04	72.28
Acq + wheat	3	100	231.584	67.45
Acq + wheat	5	Before SVD	31.99	68.88
Acq + wheat	5	5	76.95	62.94
Acq + wheat	5	10	93.101	67.73
Acq + wheat	5	20	163.92	67.73
Acq + wheat	5	50	205.06	69.15
Acq + wheat	5	100	232.44	69.08

4.3 Discussion

From the results above we can see we got better accuracy in clustering after performing LSA which strengthens our earlier hypothesis. In addition to this we also see that accuracy increases with the increase in number of dimensions used. However this is not always the case. For e.g. accuracy decreased from 72.36% to 69.52% when we increased the dimensions of data1 from 50 to 100 for 5 clusters. This suggests that when finding 5 clusters for data1 dimensions between 20 to 50 works better.

When evaluating number of clusters for accuracy we found that when the data is clustered in the number of clusters that are actually present in the data, accuracy is better. For e.g. in data1 we have 3 topics thus there are 3 logical clusters in the data. Accuracy was high in all such cases when the number of logical clusters was equal to the number of clusters we were trying to make in our experiment.

Therefore we can also say from our results that increasing the number of clusters decreased accuracy. Overall, an optimum range of 20 to 50 features gave better results for data1 whereas for data2 we got better results for even a smaller number of features.

5. Most Representative words

5.1 Finding Most representative words

Most representative words for a concept are the words that best describe that concept. These words are the most important words for that concept.

How to find most representative words for a concept in R:

The V matrix after SVD is a matrix which gives you importance of a word to a concept, if your matrix is of the type document-term matrix (for term-document matrix U will give this importance values). The code used is below:

```
wordlist <- svd$v [, i ]
rownames(wordlist) <- colnames(dtm)
sorted.wordlist <-sort(abs(wordlist), decreasing = TRUE)[1:100]
sorted.wordlist
```

Where, i is the index for i'th concept, dtm is the document term matrix, svd$v is the v matrix of the SVD object 'svd'.

5.2 Motivation for finding Most representative words

Concepts are mental structures. Words and phrases are the linguistic representatives of concepts. Due to the inherent conciseness of natural language, words can represent multiple concepts and different words may represent the same or very similar concepts.

Most representative words help us in understanding a concept. It gives us a perspective of the context of a concept. By finding the most important words we can see if a concept belongs to a certain group. Using these words we can also compare one concept to the other and if they are similar or different to each other. We can also represent concepts with words that have high relevance to that concept.

Below are the 10 **Most representative words** for first five concepts in the order of their importance.

Concept 1: said, pct, year, bank, billion, company, share, oil, rate, last

Concept 2: mln, said, dlrs, loss, cts, profit, shr, trade, market, oper

Concept 3: bank, rate, offer, oil, wheat, deficit, growth, rose, forecast, stock

Concept 4: billion, share, wheat, bank, pct, cts, harvest, crop, grain, net

Concept 5: billion, loss, mln, oil, cts, company, central, share, year, money

Other than finding the most representative words we can also find the most frequent words in a cluster or in a topic. There are the terms that appear in that topic the highest. These terms can then be represented with a word cloud.

Word clouds can be useful to visually distinguish which terms appear more frequently in a topic. The bigger the size of the word in a word cloud the higher the frequency of that word in that topic. Below is an example of a word cloud that represents most frequent terms in the topic 'crude'.

Fig 25. Word cloud for most frequent terms in topic ‘crude’

Here are some more word clouds for the most representative words for the first five concepts in the concept space.

5.4 Most Representative words

In our result of the most representative words we see that concept 1 has many words that relate to finance. In fact even concept 2, 3 and 5 too have words related to finance. However, the words are not the same. Concept 2 seems more closely related to shares and trading whereas concept 5 is more closely related to losses and money.

Concept 4 has a large number of words that belong to agriculture. It may represent the topic wheat, grain or other agriculture related topic.

Majority of the words come from finances and money and this may simply be because of large number of documents for acq and earn in our dataset.

Conclusion

SVD is an effective technique to achieve latent semantic analysis. The decomposition brings out the hidden features in a document term matrix which can be used to analyze the dataset and make concrete conclusions. The importance of a word to a feature and the importance of a feature to a document can be calculated by multiplying the term feature matrices and feature document matrices with the diagonal feature matrix respectively.

SVD can not only be used for LSA but also to perform multi-dimensional scaling. After these experiments we learned how to apply SVD for LSA effectively. We saw that after computing SVD the clusters of the documents and terms became clearer and the overlapping of words and documents could be removed to get better and disjoint clusters. This happens due to dimensionality reduction in the decomposed matrices as compared to the original document term matrices. This works best for large data set because we can choose only to use the most important features thus cutting out unnecessary overload of performing calculations on high dimensional data.

The relevance of a term to a document and vice-versa (studied through plotting of clusters) before applying SVD is not as intuitive and informative as compared to after performing SVD. It seems, after going through tutorials of SVD and class examples when making a query to find the relevance of a set of terms with respect to a set of documents using the decomposed eigenvalue matrices the results in predicting the importance will be more accurate and precise when compared to not using SVD.

Matrix Exercises

Part 2

1. What is the effect of this transformation if a term occurs in one document? In every document?

Answer: Using the formula for this transformation, the words that occur in every document have score 0, while words that occur in precisely one document have weight log(m).

2. What might be the purpose of this transformation?
   Answer: This means that frequent words like "the" is almost guaranteed to have weight 0, which means that its presence do not provide any insight as to the content of the training documents, while relatively rarer words can distinguish one sample from another.

Part 3

1. Explain what is $M*M^T$?
   Answer: The entry M_{ij} in the resulting matrix represent the un-normalized cosine similarity score between attribute i and attribute j, because it is the dot product between the two attribute vectors.

2. Explain what is M^T*M?
   Answer: The entry M_{ij} in the resulting matrix represent the un-normalized similarity score between movie i and movie j, because it is the dot product between the two movie vectors.

Part 4

The singular vectors obtained after singular value decomposition of the movie matrix can be used to project ratings from a new user in the concept space as follows:

After applying SVD on a movie matrix M we get three matrices S, E and U where S denotes attribute to concept similarity, E is the eigen vector matrix of concepts and U is the movie similarity matrix.

When a rating vector from a new user comes we can project it on the concept space by multiplying the rating vector to the movie similarity matrix. The resulting matrix will tell us what kind of movie the new user would like and we can recommend similar movies.

$$[5 \quad 0 \quad 0 \quad 0 \quad 0] \begin{bmatrix} 0.56 & 0.12 \\ 0.59 & -0.02 \\ 0.56 & 0.12 \\ 0.09 & -0.69 \\ 0.09 & -0.69 \end{bmatrix} = [2.8 \quad 0.6]$$

Similar procedure can be followed to represent a new movie in the concept space By multiplying the new movie vector with the attribute-concept similarity matrix. The resulting matrix will tell us what space does the movie conforms to.

$$[0 \quad 5 \quad 0 \quad 5 \quad 3 \quad 1 \quad 0]\begin{bmatrix}0.13 & 0.02\\ 0.41 & 0.07\\ 0.55 & 0.09\\ 0.68 & 0.11\\ 0.15 & -0.59\\ 0.07 & -0.73\\ 0.07 & -0.29\end{bmatrix} = [5.97 \quad -1.59]$$

The new movie is a Sci-fi movie in this case.

4. The matrix M*M' is a symmetric matrix where as M itself is an nxm matrix

Svd(M)=UxDxV'

Thus we can say M*M' = [UxDxV']*[UxDxV']' = [UxDxV'xVxD'xU']

= [UxDxD'xU']

Let's say H=DxD' (conceptually DxD' is also a feature matrix) and M*M'=B

Therefore M*M'= UxHxU'=B

Now applying eigenvalue decomposition on B we get

B=WZW'= UxHxU'

- WZW'= UxHxU' hence we can say W=U

Similarly M'*M=VxHxV' and M'*M =KxYxK'

Hence we can say V=K for SVD of M and eigenvalue decomposition of M*'M

Movie Data Analysis

Data:

The dataset we are using are movie ratings given by users. This data is collected by the movieLens Group and can be downloaded here: http://grouplens.org/datasets/movielens/ Out of the many datasets they have available on their website we are going to use the smaller 100k ratings dataset.

Reading the data and preprocessing

We read the file u1.base as a table using the 'read.table()' function by passing the complete location of the file as a parameter.

The data we have is in a tabular form with the following columns: *<user ID>, <Movie ID>, <rating>* and *<timestamp>*. We remove the *<timestamp>* column because we do not need it for our analysis.
We then build a user-movie matrix to compute SVD with users as rows and movies as columns. The u1.base file is a training set with 80% of the data (80000 ratings of 1682 movies by 943 users) and the rest 20% is in u1.test file.
When we create the sparse rating matrix the total elements in the data frame become 1586126 which is very large (12Mb). We reduce the number of users to reduce the data. I reduced the original 80000 to 39987 (getting exactly 534 users and 1618 movies) in our experiments.

Clustering before computing SVD

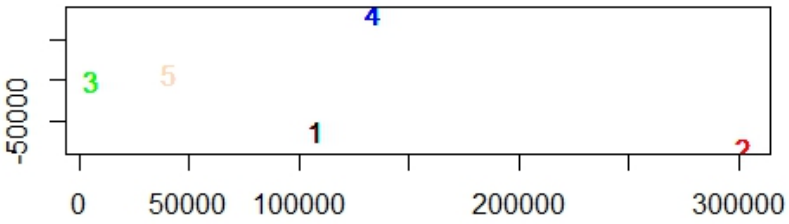

Fig 26. Clustering movies with 5 clusters

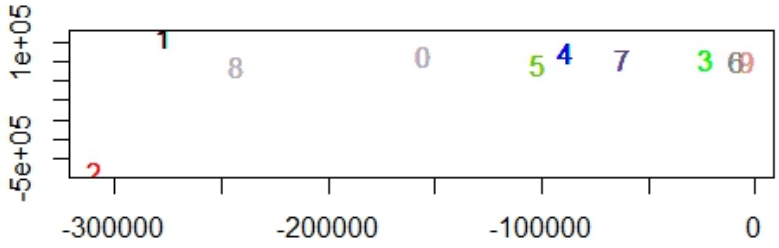

Fig 27. Clustering movies with 10 clusters

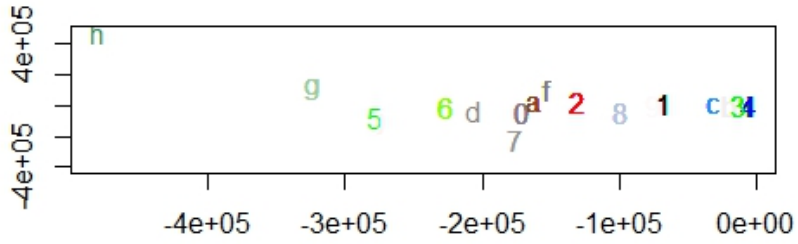

Fig 28. Clustering movies with 20 clusters

Unlike the plots in our first experiment these graphs do not help in any initial estimation of the logical number clusters that are present in the dataset. Since we do not have labels for this dataset this makes it a good example of un-supervised learning. Let's plot an SSE vs number of clusters graph to get more insight.

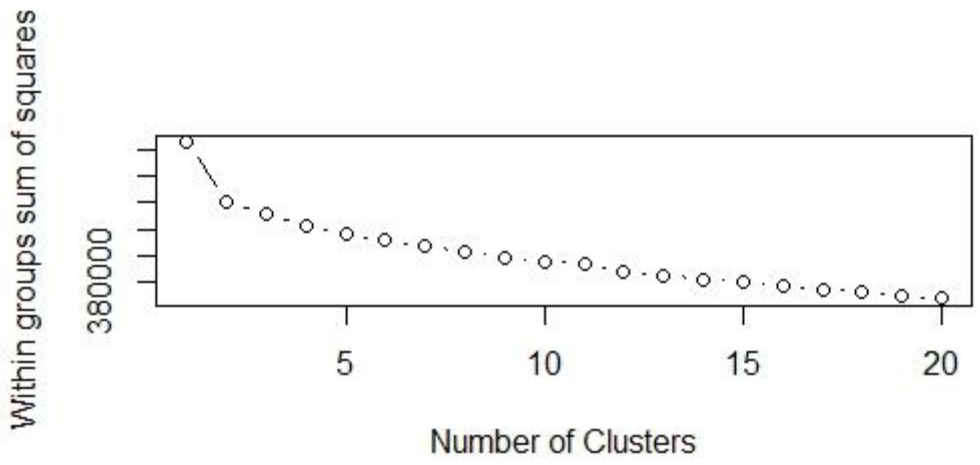

Fig 29. SSE vs number of clusters

Clustering after SVD

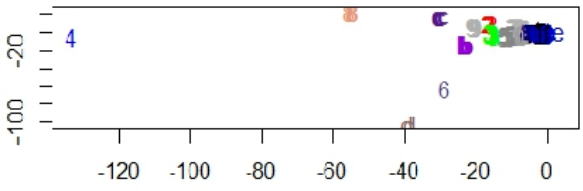

Fig 30. 300 features 15 clusters

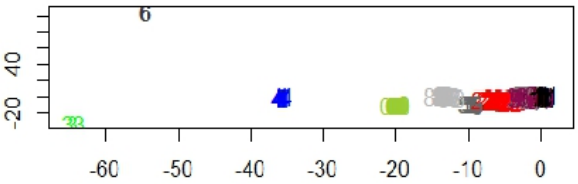

Fig 31. 300 features 10 clusters

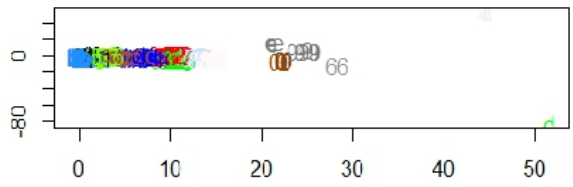

Fig 32. 200 features 15 clusters

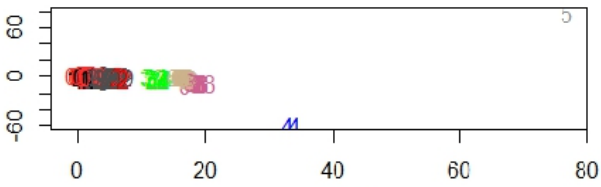

Fig 33. 200 features 10 clusters

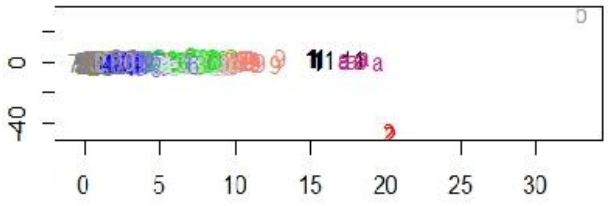

Fig 34. 100 features 15 clusters

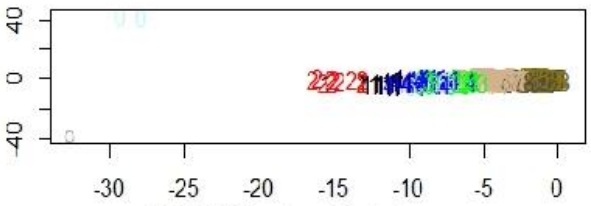

Fig 35. 100 features 10 clusters

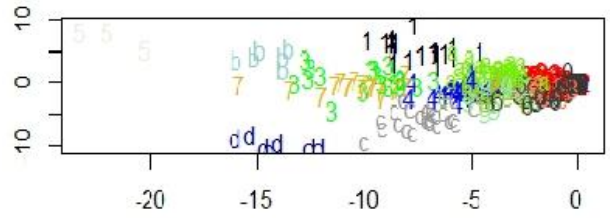

Fig 36. 50 features 15 clusters

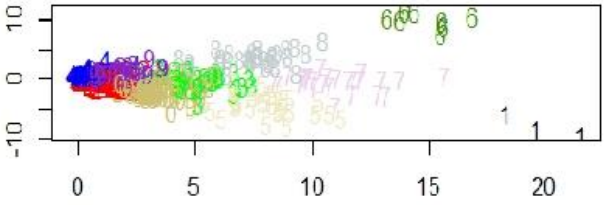

Fig 37. 50 features 10 clusters

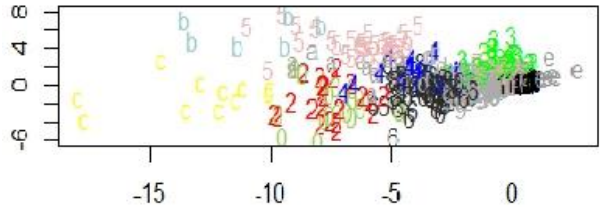

Fig 38. 10 features 15 clusters

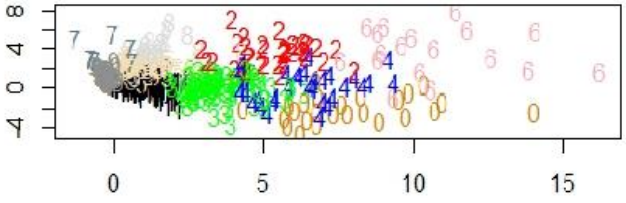

Fig 39. 10 features 10 clusters

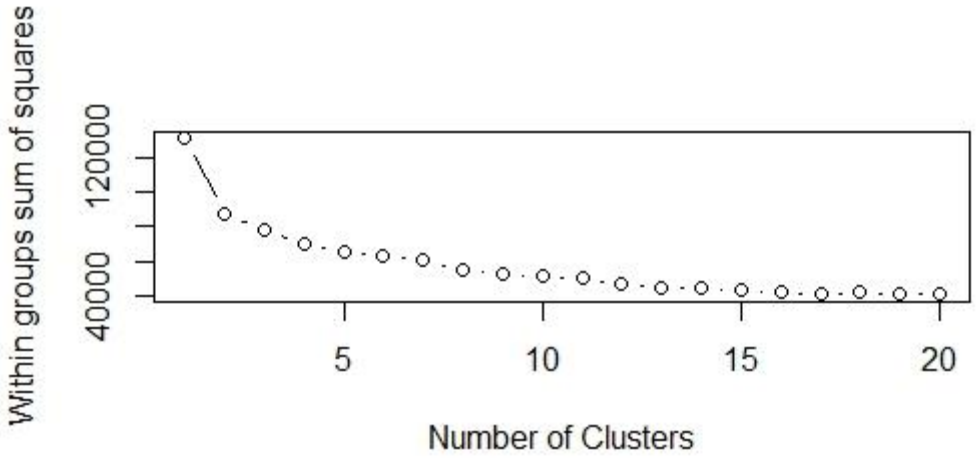

Fig 40. SSE vs Number of clusters after SVD for 10 features

List of some movies that are closest to cluster centroids for Post-Dimensional reduction KMeans with 10 number of dimensions.

Centroid #1:
Cosi (1996),Hotel de Love (1996),Broken English (1996),Hard Eight (1996),Waterworld (1995),Dangerous Beauty (1998),Turning, The (1992),Mr. Jones (1993),Beyond Rangoon (1995),Flesh and Bone (1993)
Centroid #2:
City Slickers II: The Legend of Curly's Gold (1994),Young Guns (1988),Batman Returns (1992),Coneheads (1993),Walk in the Clouds, A (1995),Star Trek IV: The Voyage Home (1986),Brothers McMullen, The (1995),Duck Soup (1933),Murder, My Sweet (1944),Ace Ventura: Pet Detective (1994)
Centroid #3:
Strange Days (1995),Sliver (1993),Ace Ventura: When Nature Calls (1995),Platoon (1986),Powder (1995),Firm, The (1993),Secret Garden, The (1993),Heavy Metal (1981),Immortal Beloved (1994),Little Rascals, The (1994)
Centroid #4:
Ed Wood (1994),Priest (1994),Star Trek: First Contact (1996),Ruby in Paradise (1993),Ben-Hur (1959),Terminator, The (1984),Chinatown (1974),Seventh Seal, The (Sjunde inseglet, Det) (1957),As Good As It Gets (1997),Candyman (1992)
Centroid #5:
Jackal, The (1997),187 (1997),Edge, The (1997),Mrs. Brown (Her Majesty, Mrs. Brown) (1997),That Darn Cat! (1997),Absolute Power (1997),Crash (1996),Ulee's Gold (1997),Air Force One (1997),Paradise Lost: The Child Murders at Robin Hood Hills (1996)
Centroid #6:
Hear My Song (1991),Akira (1988),Like Water For Chocolate (Como agua para chocolate) (1992),Addams Family Values (1993),Dead Man (1995),Geronimo: An American Legend (1993),Sex, Lies, and Videotape (1989),Replacement Killers, The (1998),Three Colors: Blue (1993),Howling, The (1981)
Centroid #7:
Emma (1996),Long Kiss Goodnight, The (1996),Ice Storm, The (1997),Up Close and Personal (1996),Raise the Red Lantern (1991),Home for the Holidays (1995),Shooter, The (1995),Lone Star (1996),Malice (1993),Garden of Finzi-Contini, The (Giardino dei Finzi-Contini, Il) (1970)
Centroid #8:
Unforgiven (1992),Empire Strikes Back, The (1980),Private Benjamin (1980),This Is Spinal Tap (1984),Evil Dead II (1987),Henry V (1989),Hour of the Pig, The (1993),Bad Boys (1995),Switchback (1997),Home Alone (1990)
Centroid #9:
2 Days in the Valley (1996),GoldenEye (1995),Independence Day (ID4) (1996),Moll Flanders (1996),Anaconda (1997),Jane Eyre (1996),Con Air (1997),Shaggy Dog, The (1959),Indian in the Cupboard, The (1995),Miserables, Les (1995)
Centroid #10:

Sunset Blvd. (1950),Giant (1956),Gigi (1958),All About Eve (1950),Notorious (1946),Adventures of Robin Hood, The (1938),My Life as a Dog (Mitt liv som hund) (1985),Somewhere in Time (1980),Miller's Crossing (1990),Laura (1944)

The list above is generated using 10 dimensions. This number is chosen because the results make sense on an intuitive level and the representative chosen are recognizable.

Conclusion

As seen in our earlier done analysis on reuters data set we see that SVD reduces the dimensions of data significantly while retaining meaningful information. It also performs better in clustering. In terms of accuracy we cannot make arguments as to which performed better due to lack of labels for the movies.

In the above result of movies that belong to one centroids are similar in nature. Centroid #9 clearly deals with action thrillers with movie titles such as Golden Eye and Independence Day. Centroid #2 can be described to affiliate with cop drama with Ace-Ventura and Batman. Centroid #6 deals with horror judging from its titles. The results match with our intuition, which means that LSA succeeded learning the concept of genre and classify movies accordingly.

We also see similar relation of SSE to the number of clusters and number of dimensions as seen in the reuters analysis. SSE increases with increase in the number of clusters and also with the increase in the number dimensions used.

The concept space produced by LSA in the movie dataset pertains to the genre of a movie or a set of genres as a movie can belong to multiple genres. In addition to generating concept space SVD can also help us in mapping new users and movies into the concept space and thus in turn can tell us movies that are similar to each other and users that have similar taste in movies.

LSA on movies data can act as a base for making movie recommendations using the similarity matrices given by SVD.