**Newsgroup dataset**

This dataset contains the information of thousands of newsgroups grouped by the category where they belong. There are 20 categories, each of them represented by a different folder. Inside each folder, there are files where each of them represents one post. All the posts are stored in text format and have the information of the sender as well as the text itself in an unstructured format. Some of the newsgroups are related with each other, usually the ones with the same prefix in the name of the newsgroup such as sci.electronics and sci.space. Both of them are related to scientific posts.

**Overview**

As shown in the Table 1, almost all of the newsgroups have similar number of documents, however the number of unique terms vary in some cases a lot. For instance, the newsgroup ‘comp.os.ms-windows.misc’ has more than 20 thousand unique words but the number of documents is very close to all of the others. So after checking some of the posts manually and realizing that this is due to technical words that maybe their frequency is close to 1, we could infer that this newsgroup have the data spread and will not have a good representation for the analysis, however we will represent it in one of the experiments to justify our hypothesis. Likewise, the newsgroup ‘talk.religion.misc’, which has a small number of documents but a high number of unique words, could be not accurate to find a relationship with others newsgroups.

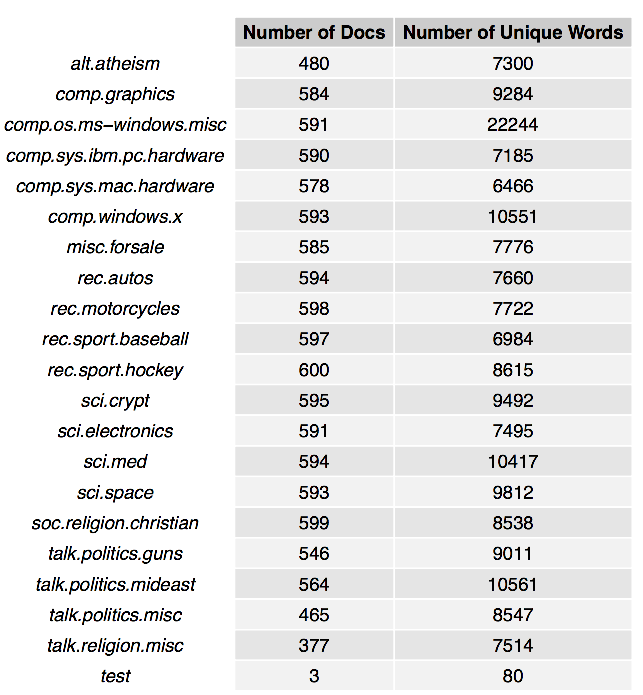


Table 1: Number of documents and unique words

The report will be mainly focused on the newsgroup with a number of documents close to the average of them, and with a number of unique terms close to the average of them.

**1st Experiment**

For our first experiment we take 3 newsgroups: ‘rec.sport.baseball’, ‘talk.politics.gun and ‘comp.sys.ibm.pc.hardware’. All of them belong to a different newsgroup, so we could infer that at least we will find 3 clusters.

First, we run the preprocessing by executing these actions over all the newsgroups in the experiment:

* Remove the words: Subject, Organization, Writes, From, Lines, Expires, NNTP-Posting-Host and Article. These words were not representing our data, as they are present in almost all the email headers.
* Convert the text to lower case.
* Remove stop words.
* Remove punctuations.
* Remove numbers.
* Stem the document.
* Strip the whitespace.
* Remove words with less than 4 letters.

Once preprocessed the data, we created a DocumentTerm matrix getting a total of 1733 rows (documents) and 17515 columns (terms).

Then, we try to find the number of clusters that fit well on the data. By using kmeans for clustering, we chose k in a range from 2 to 20 and calculated the total within groups sum of squares errors from the tf-idf matrix. Figure 1 shows us the result of the experiment.

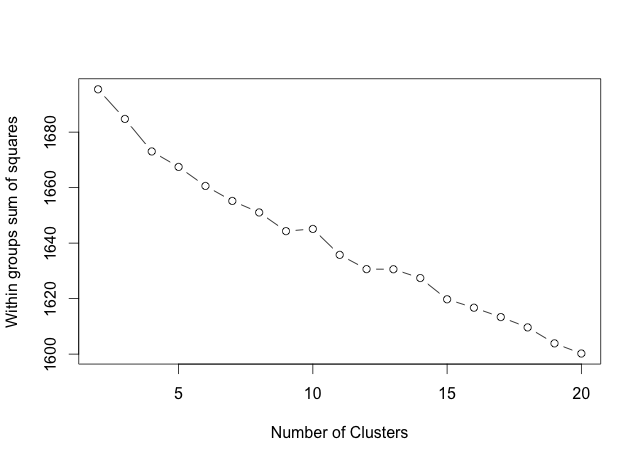
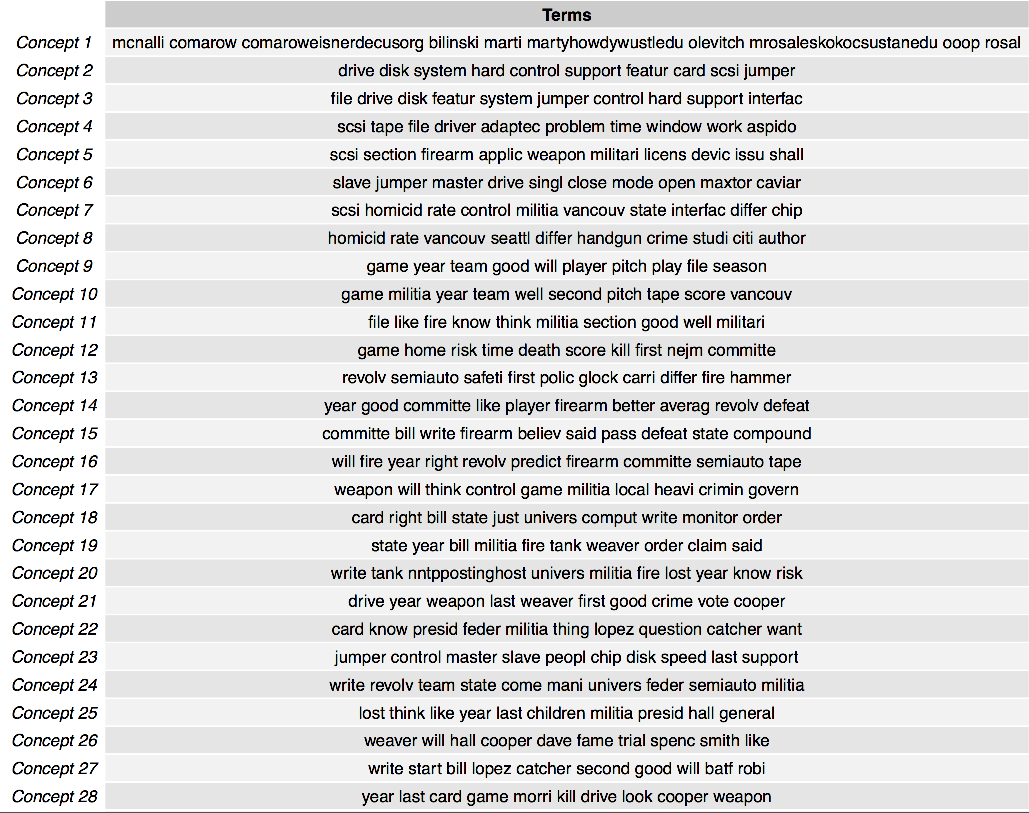


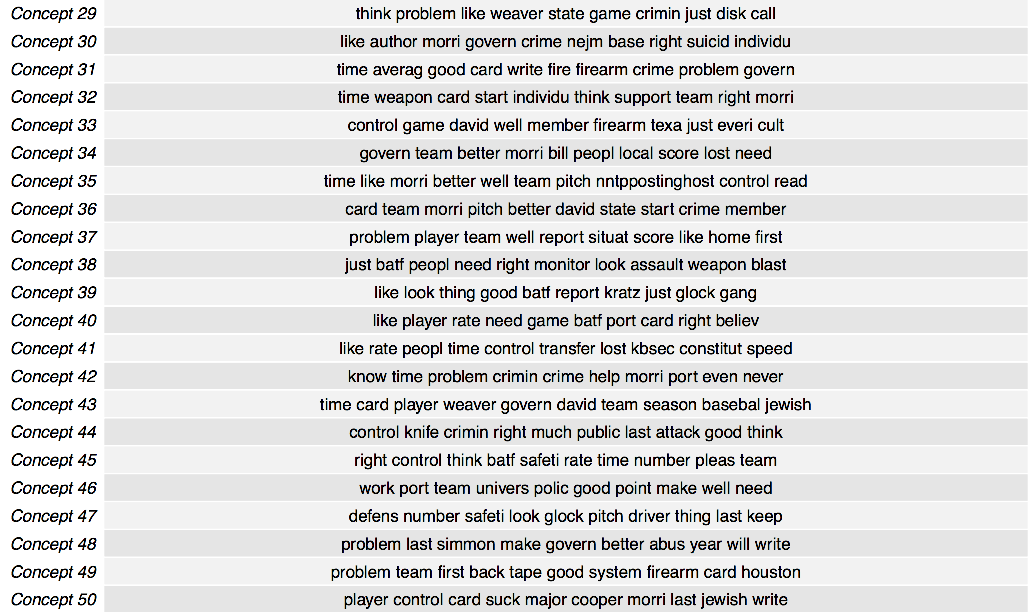
Figure 1. Tf-Idf Clustering

As we expected, the higher the value of k, the lower the SSE is too. We can also observe that in k=4, the SSE changes abruptly, so from k=4 to k=20 the slope of the line is similar, but the slope from k=2 to k=3 is different. This is called the ‘elbow’ method, and it is the one we will use for all of the kmeans evaluations in all of the experiments. In this case, we could choose k=4 as the best number of clusters for our data, however we cannot have a conclusion yet as we need more experiments to verify this result.

In order to compare with other results, we computed the SVD on the DocumentTerm matrix. The result gave us 3 matrices. The first matrix contains all of the left singular vectors, the second contains the singular values, and the third matrix contains all of the right singular vectors. In addition, each row in the first matrix represent one document and each column the concept, so in general the first matrix help us to understand the relationship between each document and the concepts. In the same way, each row in the third matrix represent a term, and each column represent a concept, so we could find if a term is strong related with one concept or another.

For our experiments, we chose 5, 10, 20 and 50 as the number of concepts to generate our LSA matrix. With this, we get the top words for each of the concepts.





Then, for each of the dimensions, we reduced our SVD to that dimension and execute the kmeans clustering over the LSA matrix generated. Figures 2-5 show us the result for each dimension.

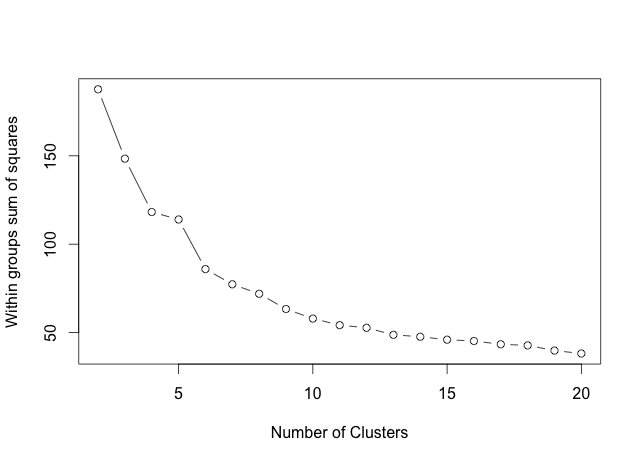


Figure 2. LSA Clustering with dimension 5

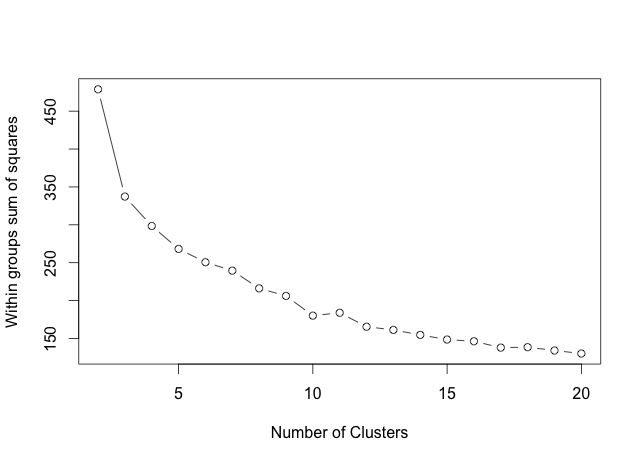


Figure 3. LSA Clustering with dimension 10

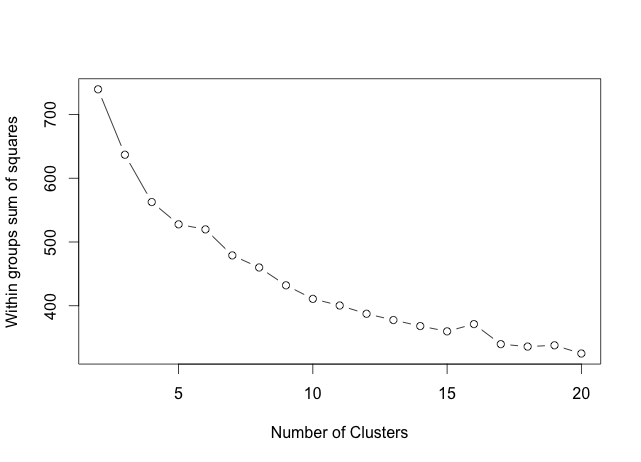


Figure 4. LSA Clustering with dimension 20

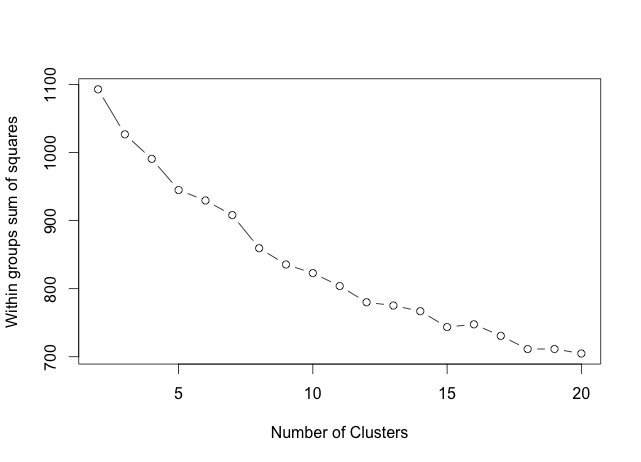


Figure 5. LSA Clustering with dimension 50

Once more, we observe that in almost all of the graphs, the slope of the line changes abruptly in k=4, especially in Figure 2, so we could say that the best number of clusters that fit our data is 4.

But the most interesting observation is that the SSE starts very low, and then it increases as we increase the dimension of the concepts. The SSE error for the tf-idf matrix was greater than 1600, but the SSE for LSA with dimension 5 was at most 200. Both of them predicted 4 clusters, but the one with LSA did it more efficiently, and even with the dimension of 50 for LSA, it predicted 4 clusters and the SSE was at most 1100. So in any of the cases, by using any of the dimensions, we conclude the LSA give us a better measure to distinguish one point from another in a cluster. The final observation of this experiment is that the slope of the line in LSA decreases as we increase the dimensions. In k=2 is really easy to observe the number of clusters required for the dataset, but as we increase k, it becomes more difficult to distinguish visually. At the end, the graph more complicated to distinguish the number of clusters visually, is the tf-idf matrix.

**2st Experiment**

For our second experiment we take 2 newsgroups: ‘comp.os.ms-windows.misc’ and ‘talk.religion.misc’. All of them belong to a different newsgroup, so we could infer that at least we will find 2 clusters. In this case we took the two newsgroups that are in the boundaries with the highest number of unique terms and with the lowest number of documents.

First, we run the preprocessing by executing these actions over all the newsgroups in the experiment:

* Remove the words: Subject, Organization, Writes, From, Lines, Expires, NNTP-Posting-Host and Article.
* Convert the text to lower case.
* Remove stop words.
* Remove punctuations.
* Remove numbers.
* Stem the document.
* Strip the whitespace.
* Remove words with less than 4 letters.

Once preprocessed the data, we created a DocumentTerm matrix getting a total of 968 rows (documents) and 27571 columns (terms).

Then, we try to find the number of clusters that fit well on the data. By using kmeans for clustering, we chose k in a range from 2 to 20 and calculated the total within groups sum of squares errors from the tf-idf matrix. Figure 6 shows us the result of the experiment.

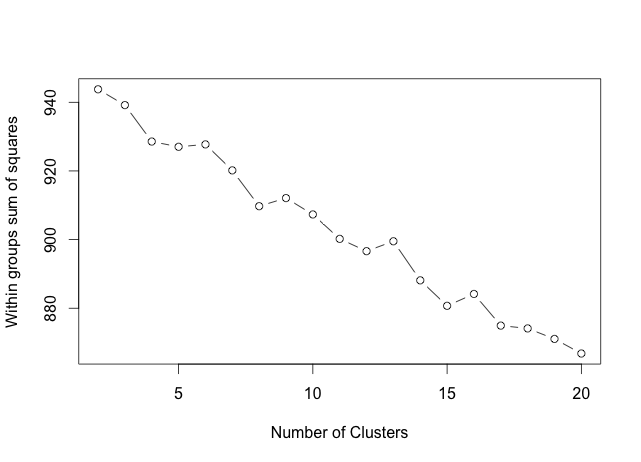


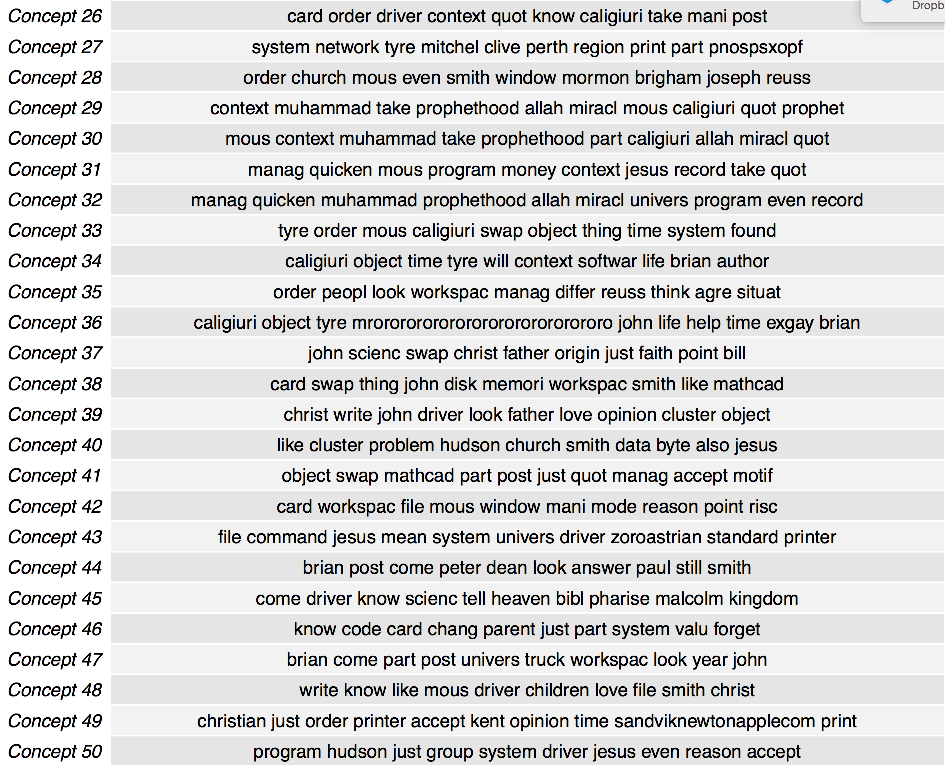
Figure 6. Tf-Idf Clustering

We can observe that according to the elbow method, k=4 is a point where SSE changes abruptly. However, k=4 is not the only one which could be chosen because k=8 seems to be a good choice too. This graph contains several changes in the slope, which make difficult to choose the correct number of cluster just by looking at this graph. For now we would say that k=4 and k=8 are the ones that fit better in our data.

In order to compare with other results, we computed the SVD on the DocumentTerm matrix. The result gave us 3 matrices. The matrix with the singular values shows 968 concepts, but we do not need all of them, but the most important ones. To reduce our document and word vectors, we keep the k singular values, while the others are set to zero. Then by multiplying the document vector, the matrix with the k singular values and the word vector, we will reduce the numbers of concepts in the LSA.

For our experiments, we chose 5, 10, 20 and 50 as the number of concepts to generate our LSA matrix. With this, we get the top words for each of the concepts.





Then, for each of the dimensions, we reduced our SVD to that dimension and execute the kmeans clustering over the LSA matrix generated. Figures 7-10 show us the result for each dimension.

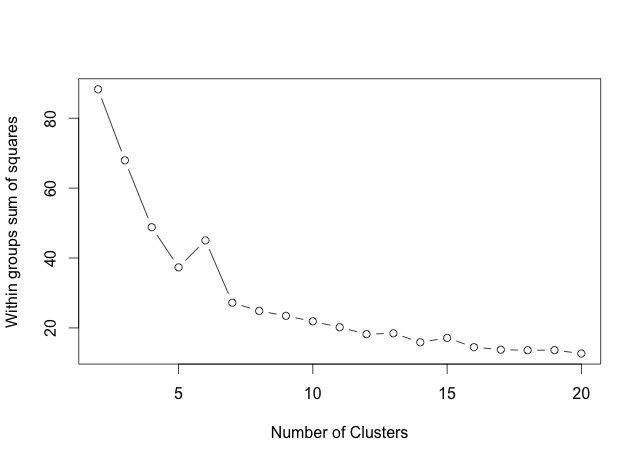


Figure 7. LSA Clustering with dimension 5

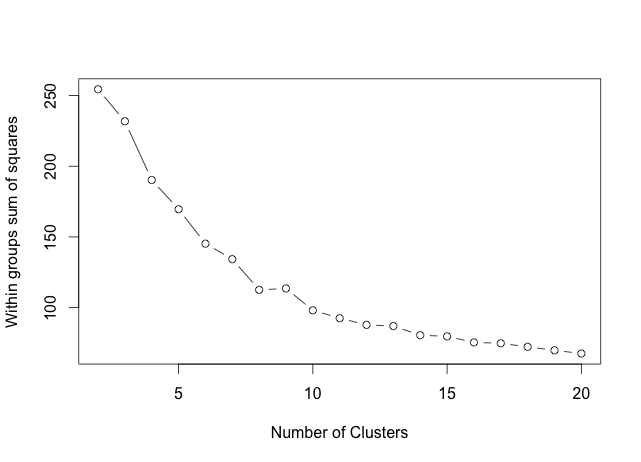


Figure 8. LSA Clustering with dimension 10

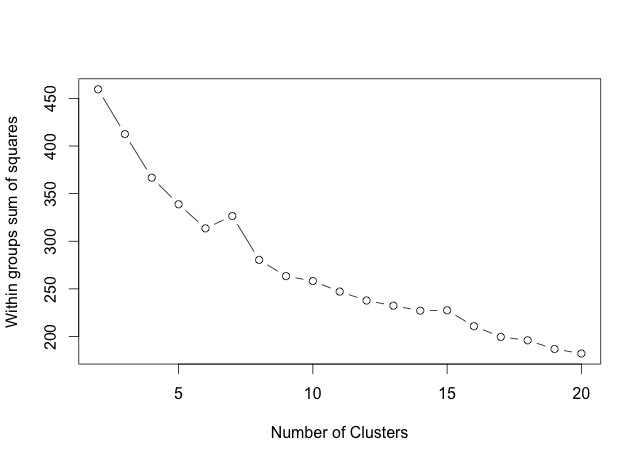


Figure 9. LSA Clustering with dimension 20

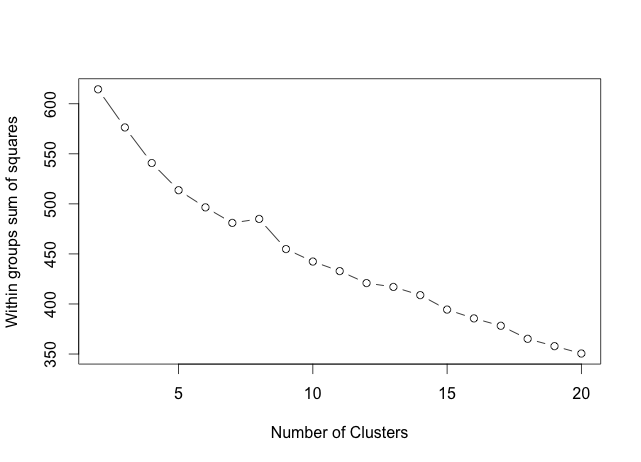


Figure 10. LSA Clustering with dimension 50

Once we observe the figures, we figure out that there is no a standard pattern well marked that say us the exact number of clusters required to fit our data. However, all of the graphs show us that for k greater than 8, the SSE starts decreasing constantly, which mean that we start doing overfitting. From k=2 to k=8 in almost all of the graphs, we can see the SSE decrease quickly, so our prediction from the tf-idf was correct. Thanks to the different experiments and visualizing the graphs, we can infer that k will be between 2 and 8, but in order to know which is the one that fits the data, we need to check manually the clusters and see the frequent words on them. This is understandable because clustering is an unsupervised method, so it requires our experience, knowledge and manual analysis.

**Reuters 21578 dataset**

This dataset has the information of different news from Reuters Ltda. This dataset was created in 1987 but was public in 1990 for research purposes in Information Retrieval. Each of the files is a newswire story and each of the folders represent a category in which the files have been assigned manually. As this data has several years and it is widely used for information retrieval problems, the text in the documents is clear with just the story itself. There are 91 categories, each of them represented by a different folder.

**Overview**

As we can observe in the Table 2, this dataset contain topics wit a small number of documents. This is because it has been manipulated manually for Reuters to have a good categorization. In general there are just 2 topics with many documents and unique words, such as ‘acq’ and ‘earn’. In order to find a relationship we need to have enough training data, which means enough documents with enough unique words. By looking manually some documents, we could figure out that there exist some documents with just one word. This will give us so much noise for our analysis and we have to be aware of that. In the following experiments we will focus in the categories with more unique words and at least 50 documents, so we could find out new relationships among them.

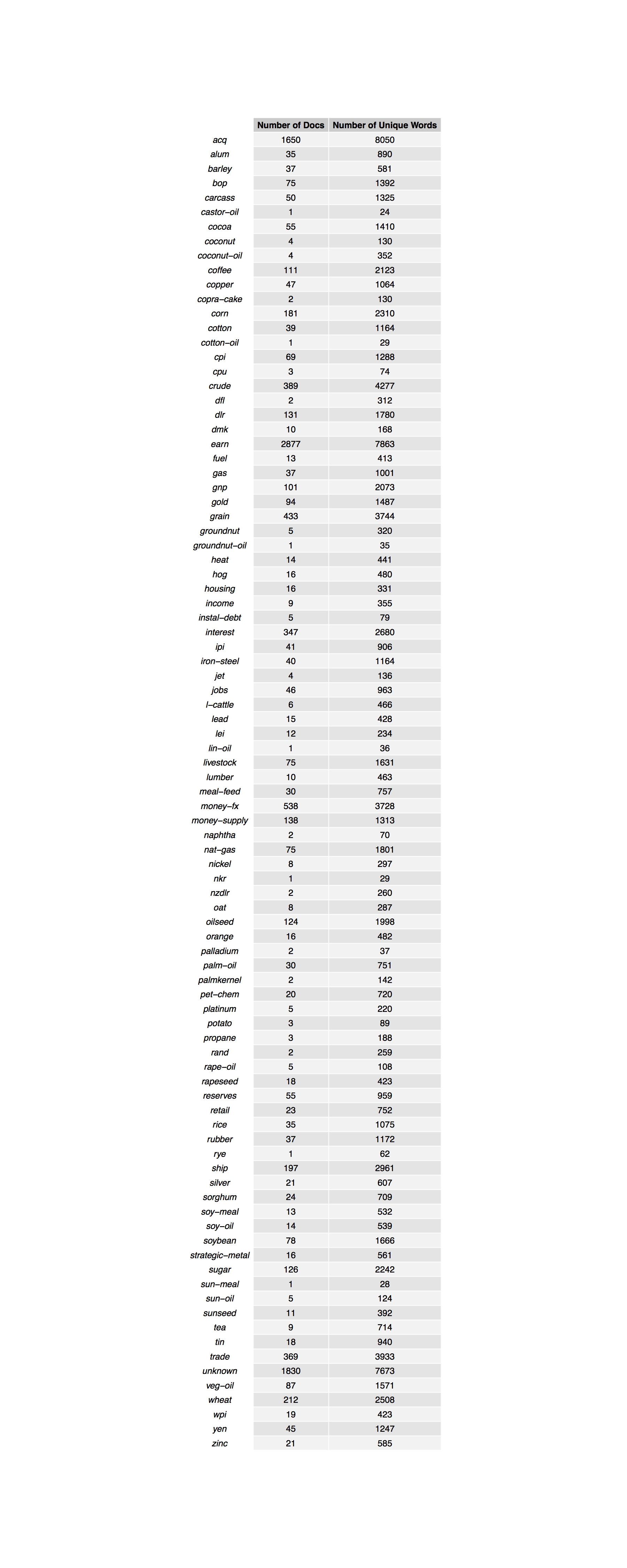
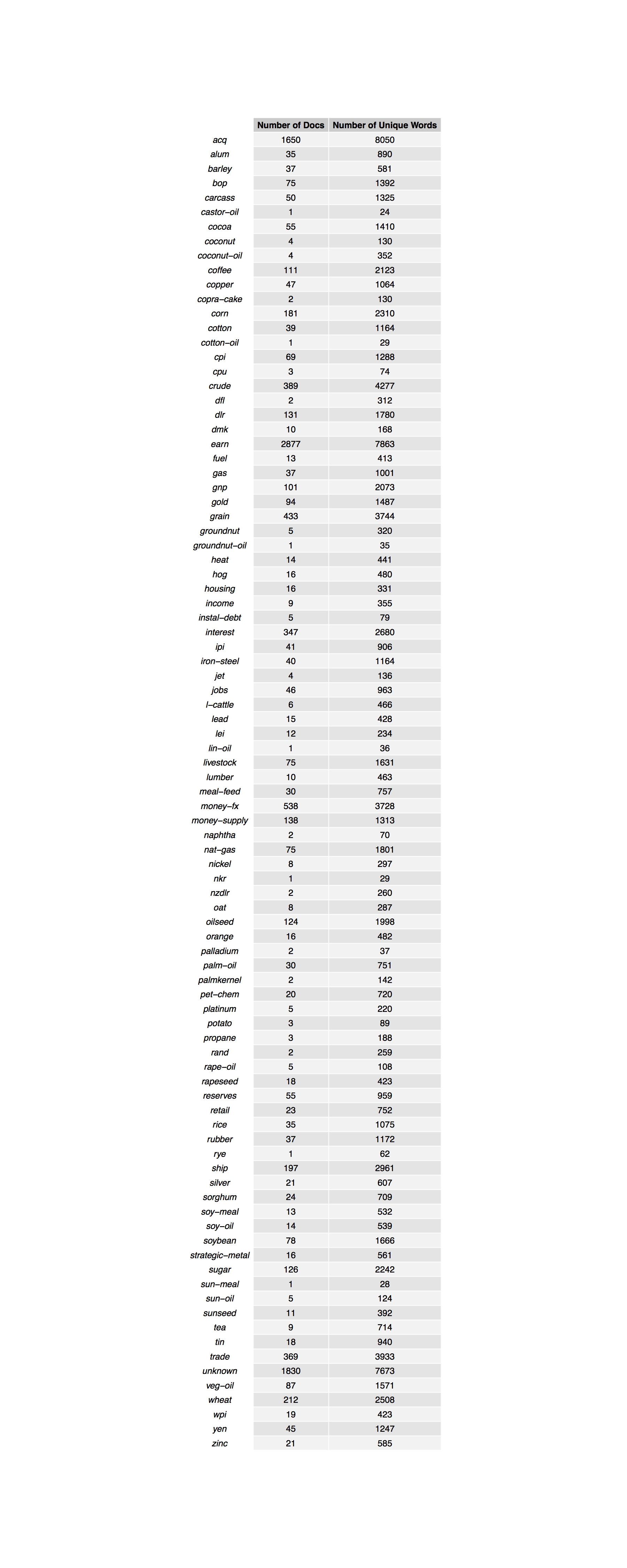


Table 2. Number of Documents and number of unique words

**1st Experiment**

For our first experiment we take 3 categories: ‘interest’, ‘gold’ and ‘money’. All of them belong to different categories. However, just by reading their names, we could infer they are related maybe in more than 3 cluster.

First, we run the preprocessing by executing these actions over all the newsgroups in the experiment:

* Convert the text to lower case.
* Remove stop words.
* Remove punctuations.
* Remove numbers.
* Stem the document.
* Strip the whitespace.
* Remove words with less than 4 letters.

Once preprocessed the data, we created a DocumentTerm matrix getting a total of 441 rows (documents) and 3329 columns (terms).

Then, we try to find the number of clusters that fit well on the data. By using kmeans for clustering, we chose k in a range from 2 to 20 and calculated the total within groups sum of squares errors from the tf-idf matrix. Figure 11 shows us the result of the experiment.

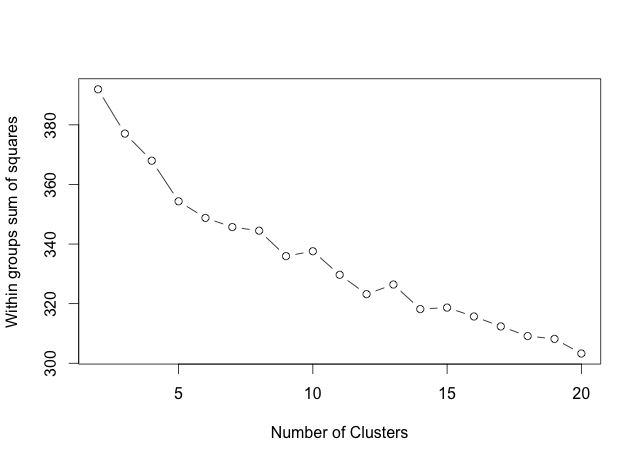
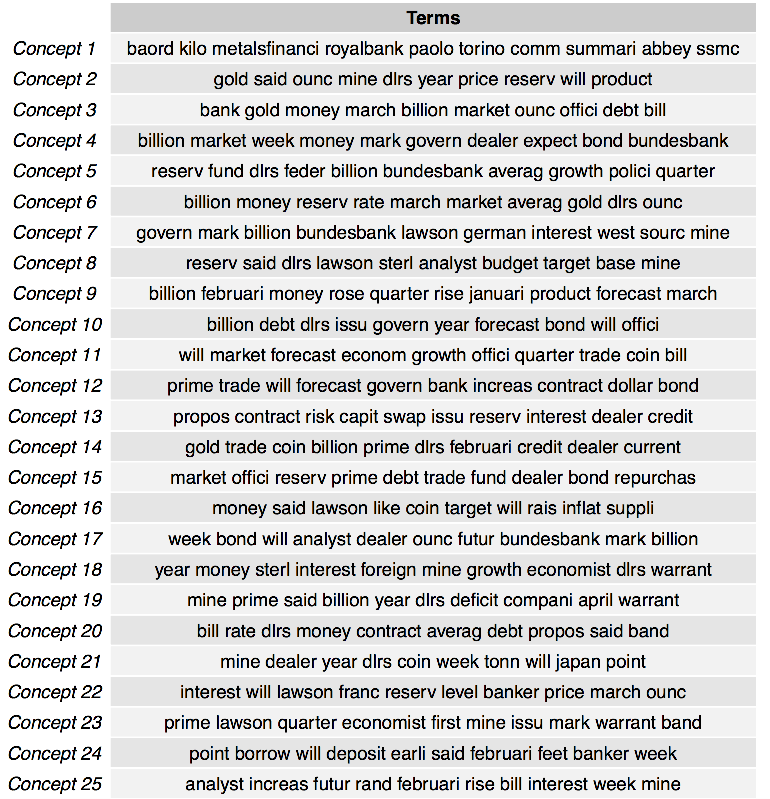


Figure 11. Tf-Idf Clustering

According to the graph, k=5 seems to be the number of clusters required for this dataset. After k=5, the SSE starts decreasing continuously until k=20 which is our last experiment. Our dataset is composed of three categories, and exactly in k=3 we can also see that the slope of the line change a little bit, which means that even tough we could divide our dataset in 3 clusters (the 3 known categories), we also could divide it into 4 or 5 clusters because they are related in some way.

In order to compare with other results, we computed the SVD on the DocumentTerm matrix. The result gave us 3 matrices. The first matrix contains the relationship between the documents and the concepts. The second matrix contains the singular values. In this experiment, the SVD retrieved 441 concepts, which is the same number of documents in our dataset. This means that our data could be spread and we could not find a better clustering than the one that we already know.

For our experiment, we chose 5, 10, 20 and 50 as the number of concepts to generate our LSA matrix. With this, we get the top words for each of the concepts.





Then, for each of the dimensions, we reduced our SVD to that dimension and execute the kmeans clustering over the LSA matrix generated. Figures 12-15 show us the result for each dimension.

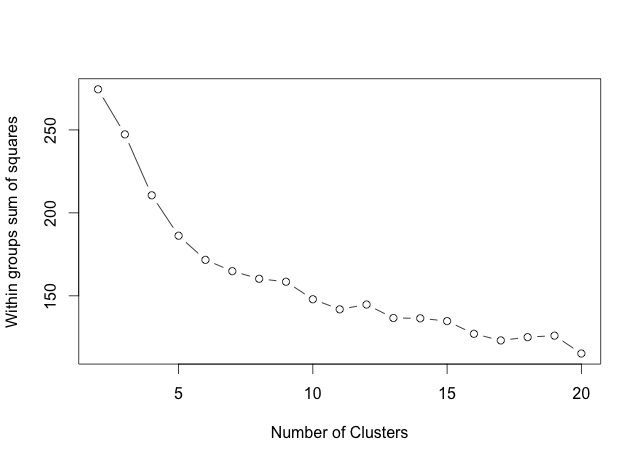


Figure 12. LSA Clustering with dimension 5

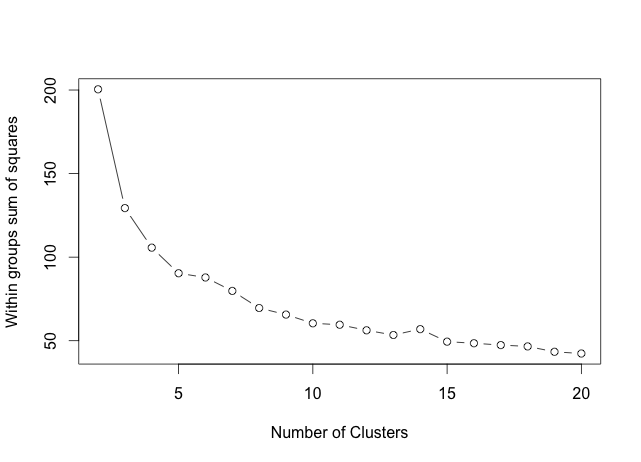


Figure 13. LSA Clustering with dimension 10

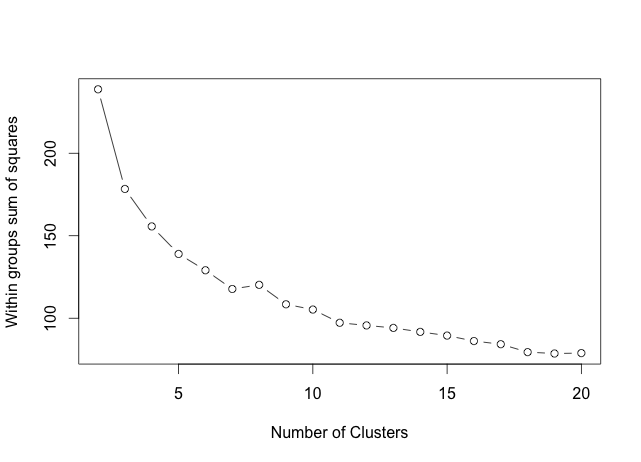


Figure 14. LSA Clustering with dimension 20

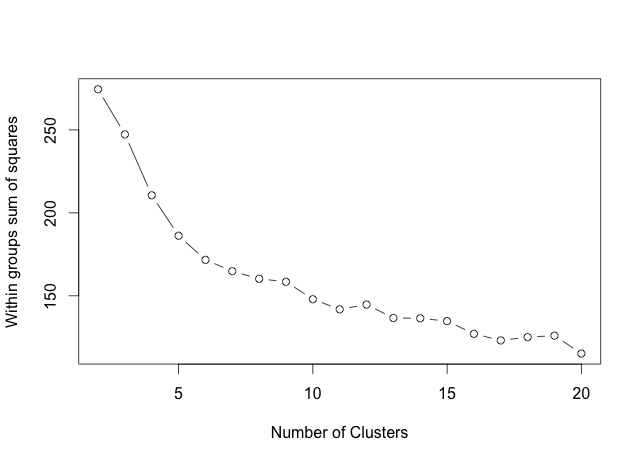


Figure 15. LSA Clustering with dimension 50

Once more, we observe that in almost all of the graphs, the slope of the line changes abruptly in k=5, especially in Figure 3, so we could say that the best number of clusters that fit our data is 5.

But the most interesting observation is that the SSE in Figure 3 with 10 dimensions is lower that the one with dimension 5, and the graphic is clearer and understandable. It seems that for this example the best number for the dimensions is 10. The SSE in the tf-idf is higher than any of the experiments with LSA, however it gave us a good perspective of what we were looking for.

**2nd Experiment**

For our second experiment we take 2 categories: ‘acq’ and ‘corn’. We do not know what is the relationship between them, however they should have at least 2 clusters well defined. This time we choose those categories because one of them has a large number of documents and unique terms, so we could find interesting results.

First, we run the preprocessing by executing these actions over all the newsgroups in the experiment:

* Convert the text to lower case.
* Remove stop words.
* Remove punctuations.
* Remove numbers.
* Stem the document.
* Strip the whitespace.
* Remove words with less than 4 letters.

Once preprocessed the data, we created a DocumentTerm matrix getting a total of 1831 rows (documents) and 8779 columns (terms).

Then, we try to find the number of clusters that fit well on the data. By using kmeans for clustering, we chose k in a range from 2 to 20 and calculated the total within groups sum of squares errors from the tf-idf matrix. Figure 16 shows us the result of the experiment.

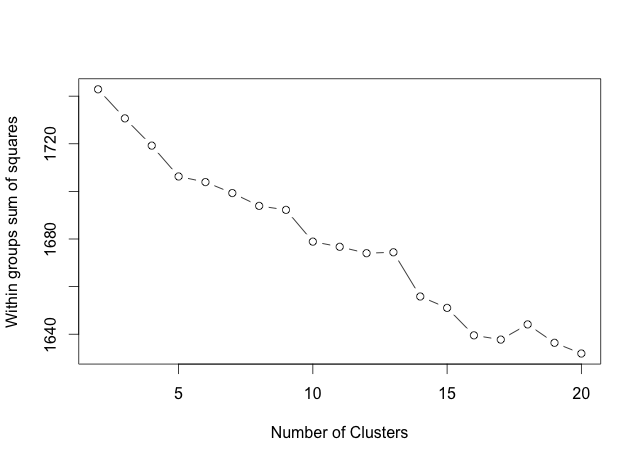
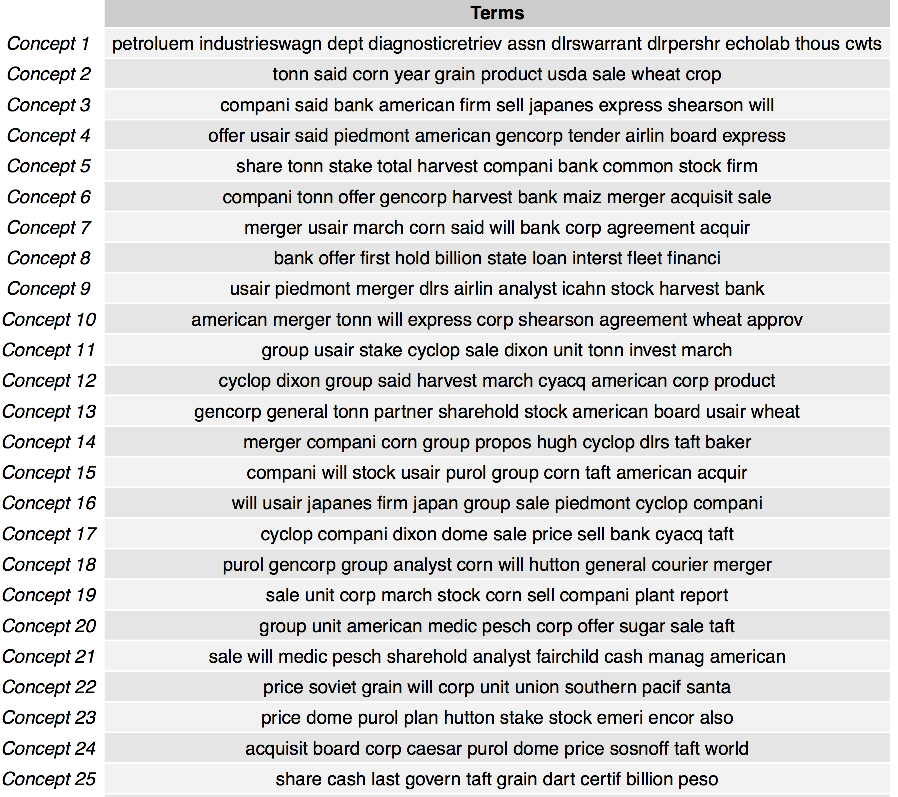


Figure 16. Tf-Idf Clustering

According to the graph, k=5 seems to be the number of clusters required for this dataset. However, it is kind of confusing because there exist other points that could be interpreted as good number of clusters too. From k=5 to k=9, SSE seems to decrease continually, but in k=10, it changes abruptly, so we could think that k=10 is another point according to the elbow method.

In order to compare with other results, we computed the SVD on the DocumentTerm matrix. The result gave us 3 matrices. The first matrix contains the relationship between the documents and the concepts. The second matrix contains the singular values. In this experiment, the SVD retrieved 1831 concepts, which is the same number of documents in our dataset. This means that our data could be spread and we could not find a better clustering than the one that we already know.

For our experiment, we chose 5, 10, 20 and 50 as the number of concepts to generate our LSA matrix. With this, we get the top words for each of the concepts.





Then, for each of the dimensions, we reduced our SVD to that dimension and execute the kmeans clustering over the LSA matrix generated. Figures 17-20 show us the result for each dimension.

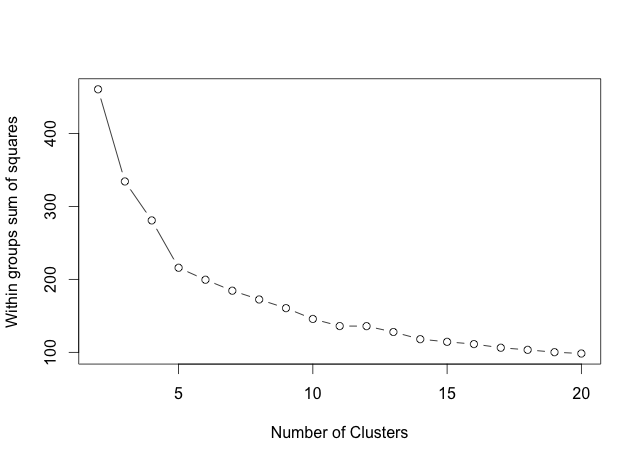


Figure 17. LSA Clustering with dimension 5

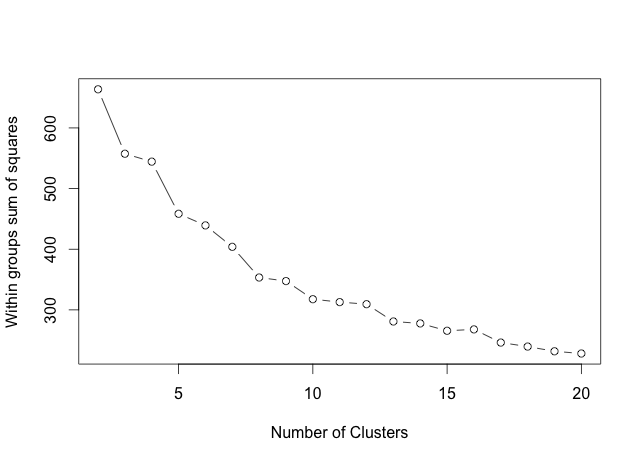


Figure 18. LSA Clustering with dimension 10

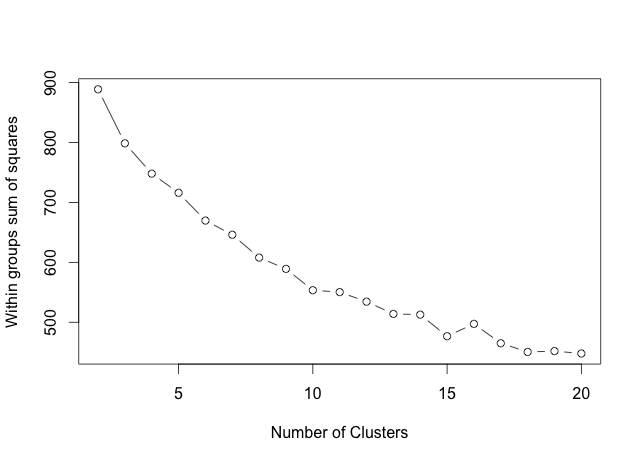


Figure 19. LSA Clustering with dimension 20

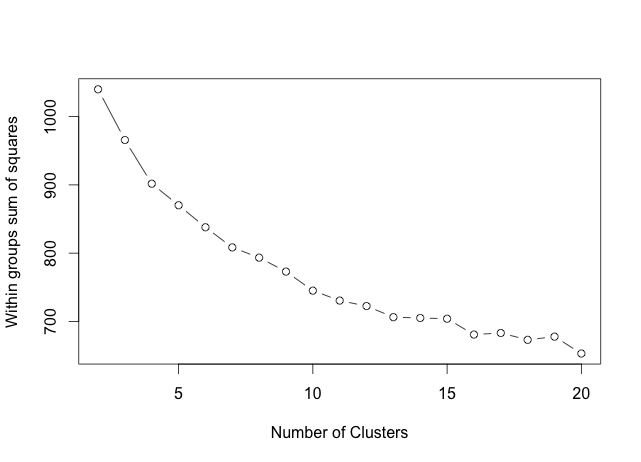


Figure 20. LSA Clustering with dimension 50

Once more, we observe that in almost all of the graphs, the slope of the line changes abruptly in k=5, especially in Figure 3, so we could say that the best number of clusters that fit our data is 5.

Even if we observe the top words per concept, we can figure out that the words in the first 5 concepts are not repeated, but from the concept 6 they start repeating. The more concepts we take, the more times a word will be repeated. This is the reason that the graphic with dimension 5 define very well our number of clusters required. Even the dimension 10 we still could conclude that we need 5 clusters for our data, however the higher is the dimension, the more difficult to visualize the optimal number of clusters.

**Conclusion**

In order to have a better visualization of the number of clusters required in our datasets, we had to apply LSA and try with different dimensions until get the one that represent our data in a better way. In this report, as our main data mining technique was clustering, we found out that tf-idf is good but not enough to give us a complete understanding of the number of clusters required for our data.

To find the best number of clusters we apply the elbow method by visualizing the point k in which the slope of the line changed abruptly. Although it is not the best way to determine the number of clusters, it gives us a good idea about it.

Usually, as we increment the dimension of the LSA, the graphic was becoming more and more similar to the one of the tf-idf. This is because in tf-idf we just use the weights, but the relations between which document belongs to a specific cluster is never weighted. On the other hand, LSA set up the concepts to make stronger the links between documents. If we increase too many the concepts, we are basically doing what tf-idf is doing too. This is the reason why we need to know how to choose the best dimension with the Frobenius norm.

LSA helped us in most of the cases, however sometimes it also fails. In the second experiment of the newsgroup dataset, the tf-idf did not show an intuitive graph to determine the number of clusters, so we tried with LSA with different dimensions and the different number of clusters. At the end, LSA was not able to show us a common result among the different dimensions. It was understandable by looking at the first words for each of the concepts. Here we could notice that exists a strong overlapping over the concepts, making difficult to choose a single number of clusters. However, LSA was useful to discard number of clusters greater than 8. Then, to select the best number of clusters between 2 and 8, is required the knowledge, experience and analysis of the data analyst. As clustering is a unsupervised method, it is normal to have these cases.