Clustering Text Documents using Fuzzy-C Means Clustering

By: Arjhun Hariharan (.36) and Varun Mohankumar (.7)

Group Section

1. Abstract

In this project, we attempt to cluster text documents using the Fuzzy C Means clustering technique. Subsets of Reuters 21578 and the Ohsumed data were used for this project. This clustering technique is a form of clustering where each data point (document in this case) can belong to more than one cluster. These clusters were identified using the Euclidean distance measure. The bag of words model and TD-IDF scores were used to represent documents as vectors.

A subset of the Reuters-21578 and the Ohsumed data were used for this project.

In both datasets it was observed that Fuzzy-C means clustering is capable of tackling multi-class problems by using appropriate tuning parameters. Fuzzy-C means algorithm gives the degree of association to each cluster which is advantageous to multi-class problems unlike K-means.

2. Introduction

2.1 Problem Statement

An important technique in organization of data is clustering. It helps to group similar documents together and separate dissimilar documents as far away from each other. Information retrieval is an important area of application for clustering of text documents.

A brief review of the literature suggests that one of the most commonly used algorithm for clustering text documents is the K Means Clustering algorithm. Though it is very powerful despite its simplicity, it's specific relevance in the field of text classification could be debatable because of the property of the algorithm that each data point will be a part of one and only one cluster. In the case of text documents, assuming that each cluster of documents would be completely related to each other and that the documents from different clusters would be completely unrelated would be a gross simplification.

This motivates the use of the Fuzzy C Means Clustering algorithm. In this algorithm, unlike the K means algorithm, each data point does not have to belong to a single cluster. Instead they could belong to a number of clustering in varying levels given by their membership function.

In this project, we attempt to implement the Fuzzy C Means clustering algorithm and use it to cluster documents in two text corpuses and analyze the results of the clustering. The bag of words model and TD-IDF scores were used to represent documents as vectors. A combination of existing packages and custom functions were used for text processing.

2.2 Literature Review

Text document similarity has been of interest in the recent past and a number of researches and papers have been reported. A few of them within our scope is listed here.

In [1], the author Huang, uses the K Means clustering algorithm to analyze and compare the effectiveness of a number of similarity measures including Euclidean Distance, Pearson Correlation Coefficient and KL Divergence. This analysis was done for 7 corpuses.

In [2], a number of clustering techniques including agglomerative hierarchical clustering, standard K means and bisecting K means clustering algorithms were compared for text classification on a number of corpuses.

The Fuzzy C Means Clustering algorithm was used for classifying the Reuters 21578 dataset in [3]. K Nearest neighbors was also used in this paper. The TF-IDF metric was used for ranking documents.

Moertini [4] aims to introduce the most representative algorithms used in off-line mode that apply crisp or fuzzy approach. The algorithms are K-Means, Fuzzy C-Means (FCM), Mountain, Subtractive and psFCM. The implementation of two of the algorithms using Matlab is provided in the appendix.

Mohammad et al. [5] aims to give a review of some popular fuzzy cluster validity indices is given. An index that is based on the generalization of silhouettes to fuzzy partitions is compared with the reviewed indices in conjunction with fuzzy c-means clustering.

2.3 Software/Packages

The entire project was done in Python and the PySpark API [6]. The Databricks [7] platform was used with a 6GB cluster (part of the free community edition).

The packages/libraries used are:

1. Pandas (data handling)
2. NumPy (data handling)
3. NLTK (for stemming)
4. Scikit-fuzzy (for off the shelf fuzzy C Means algorithm)
5. Scikit-learn (for PCA)
6. Matplotlib (for plots)
7. Scipy (K means clustering algorithm)

2.4 Report Outline

In section 3 of this report, we describe the datasets we have used for this study, the tools and the methodology used for data preprocessing and exploratory data analyses. Then, in section 4, we discuss the overall methodology used for this study. The next section will focus on the results and the analyses surrounding the results. Finally, the last section will have the list of references we have used during the course of this project.

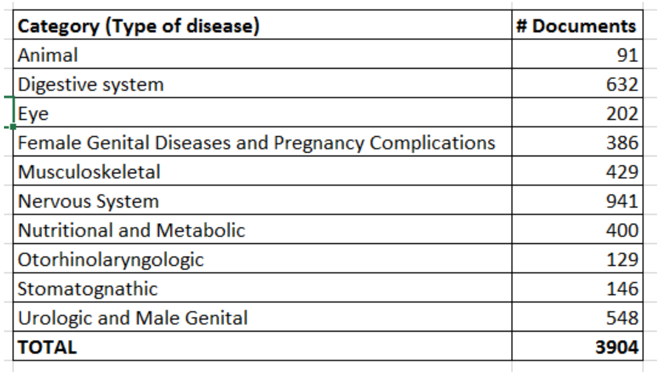
3. Dataset(s)

3.1 Description

Ohsumed:

The OHSUMED test collection is a set of 348,566 references from MEDLINE, the on-line medical information database, consisting of titles and/or abstracts from 270 medical journals over a five-year period (1987-1991). [10]

The first 20,000 abstracts of the year 1991 was the data considered for this project. It was downloaded from [11]. It consists of 23 categories of cardiovascular diseases. [11] Again, owing to the constraints in computation, this dataset was further reduced to a total of 3904 documents from 10 disease categories. The split up of these is shown below.



Category (Type of disease)	**# Documents**
Animal	91
Digestive system	632
Eye	202
Female Genital Diseases and Pregnancy Complications	386
Musculoskeletal	429
Nervous System	941
Nutritional and Metabolic	400
Otorhinolaryngologic	129
Stomatognathic	146
Urologic and Male Genital	548
TOTAL	**3904**

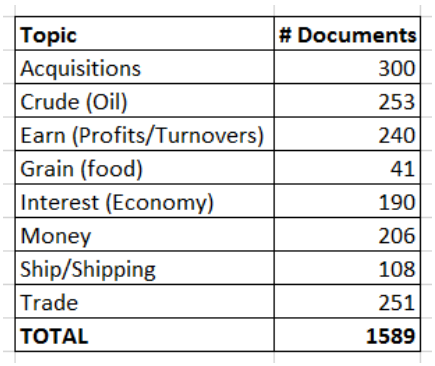
The Ohsumed data description mentions that it is a multi-label corpora and hence each document could be assigned to more than one category. This was the reason behind choosing this dataset for our study.

Reuters – 21578:

The documents in the Reuters-21578 collection appeared on the Reuters newswire in 1987. The documents were assembled and indexed with categories by personnel from Reuters Ltd. and Carnegie Group, Inc. [8]

This data is widely used for text categorization. The complete dataset contains 21578 documents. However, a condensed version of the dataset containing only 8 categories was available at Ana Cardoso Cachopo's Homepage [9]. This dataset had a total of 7674 documents. Owing to computational constraints we had, we used a reduced version of this dataset. The final dataset we used for our project had 8 topics and a total of 1589 documents.

The topics seemed to have some overlap. For example, some of the topics in the dataset were 'Money', 'Acquisitions', 'Trade' and 'Interest. As one can intuitively guess, there could potentially be a lot of overlap between the documents and hence this dataset was used for this project. Below is the split up of the number of documents in each category.

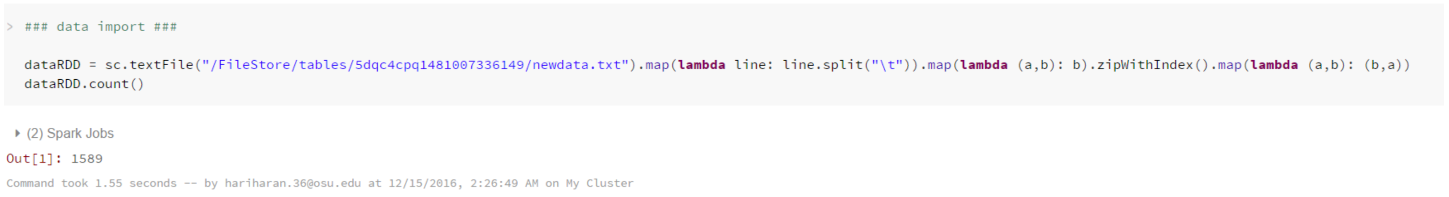


Topic	# Documents
Acquisitions	300
Crude (Oil)	253
Earn (Profits/Turnovers)	240
Grain (food)	41
Interest (Economy)	190
Money	206
Ship/Shipping	108
Trade	251
TOTAL	**1589**

3.2 Data Preprocessing

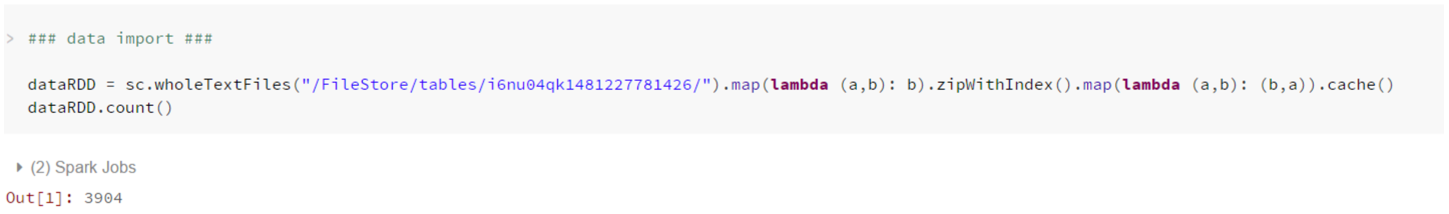
Step 1 - Data Importing:

The Reuters data that we used was obtained in a single text file, where each line had a document and in each line, the topic and the document were tab spaced. The data import code for the Reuters data is shown below.



```
> ### data import ###

  dataRDD = sc.textFile("/FileStore/tables/5dqc4cpq1481007336149/newdata.txt").map(lambda line: line.split("\t")).map(lambda (a,b): b).zipWithIndex().map(lambda (a,b): (b,a))
  dataRDD.count()

 ▸ (2) Spark Jobs
Out[1]: 1589
Command took 1.55 seconds -- by hariharan.36@osu.edu at 12/15/2016, 2:26:49 AM on My Cluster
```

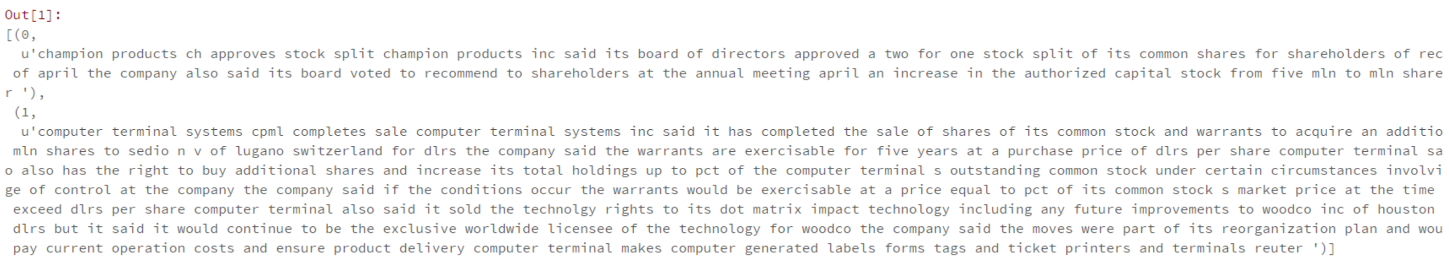
In the case of Ohsumed data, each abstract (document) was a separate file and each category was a separate folder. However, this didn't complicate the importing process much. The code that was used to import this data is shown below:



```
> ### data import ###

  dataRDD = sc.wholeTextFiles("/FileStore/tables/i6nu04qk1481227781426/").map(lambda (a,b): b).zipWithIndex().map(lambda (a,b): (b,a)).cache()
  dataRDD.count()

 ▸ (2) Spark Jobs
Out[1]: 3904
```

After import, in case of both the datasets, the RDD format was [(index1,document1), (index2, document2)…]

A small snippet of the Reuters data just after import is shown below.



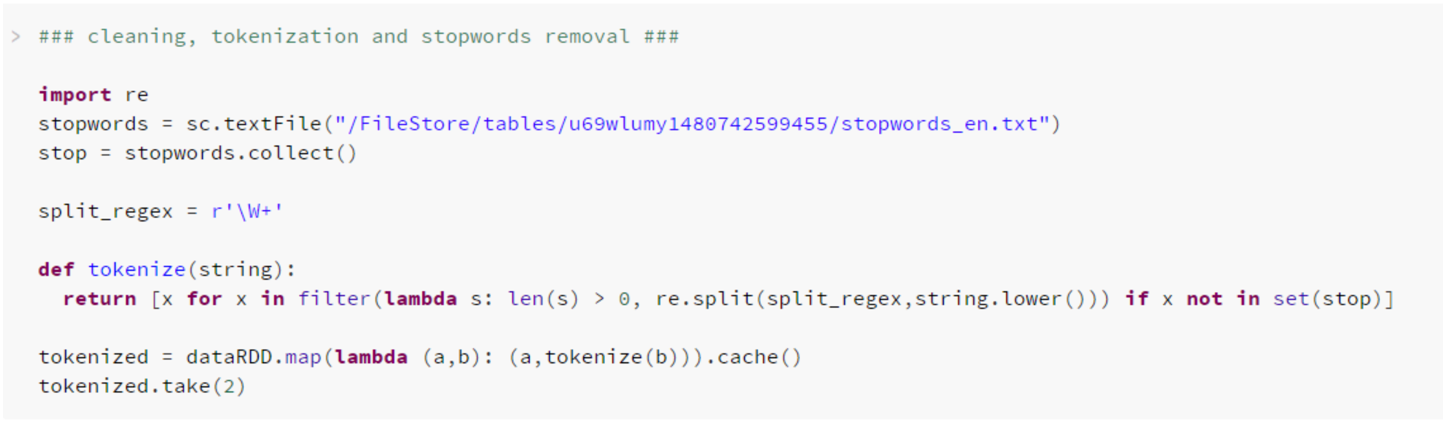
```
Out[1]:
[(0,
  u'champion products ch approves stock split champion products inc said its board of directors approved a two for one stock split of its common shares for shareholders of rec
 of april the company also said its board voted to recommend to shareholders at the annual meeting april an increase in the authorized capital stock from five mln to mln share
r '),
 (1,
  u'computer terminal systems cpml completes sale computer terminal systems inc said it has completed the sale of shares of its common stock and warrants to acquire an additio
 mln shares to sedio n v of lugano switzerland for dlrs the company said the warrants are exercisable for five years at a purchase price of dlrs per share computer terminal sa
o also has the right to buy additional shares and increase its total holdings up to pct of the computer terminal s outstanding common stock under certain circumstances involvi
ge of control at the company the company said if the conditions occur the warrants would be exercisable at a price equal to pct of its common stock s market price at the time
 exceed dlrs per share computer terminal also said it sold the technolgy rights to its dot matrix impact technology including any future improvements to woodco inc of houston
 dlrs but it said it would continue to be the exclusive worldwide licensee of the technology for woodco the company said the moves were part of its reorganization plan and wou
 pay current operation costs and ensure product delivery computer terminal makes computer generated labels forms tags and ticket printers and terminals reuter ')]
```

Step 2 - Cleaning, stop words removal and tokenization:

Once the data was imported, the next step was to clean the data (remove punctuations, unnecessary white spaces and numbers). This was done using a regular expression. After this, the documents were tokenized. In lexical analysis, tokenization is the process of breaking a stream of text up into words, phrases, symbols, or other meaningful elements called tokens. The list of tokens becomes input for further processing such as parsing or text mining. [12]

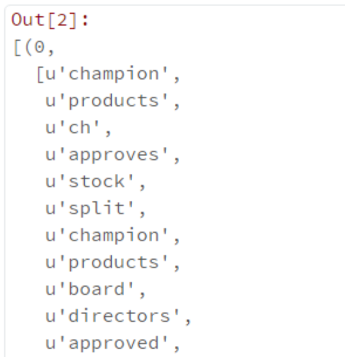
A list of around 570 stop words in English was downloaded from Kevin Bouge's website [13]. All the occurrences of these words were then removed. Stop words are words in a language that occur very frequently and hence do not add value to the analysis. Some examples of stop words include 'the', 'a', 'is', 'have' etc.

All these processes were done with custom code in Python shown below.



```
> ### cleaning, tokenization and stopwords removal ###

  import re
  stopwords = sc.textFile("/FileStore/tables/u69wlumy1480742599455/stopwords_en.txt")
  stop = stopwords.collect()

  split_regex = r'\W+'

  def tokenize(string):
    return [x for x in filter(lambda s: len(s) > 0, re.split(split_regex,string.lower())) if x not in set(stop)]

  tokenized = dataRDD.map(lambda (a,b): (a,tokenize(b))).cache()
  tokenized.take(2)
```

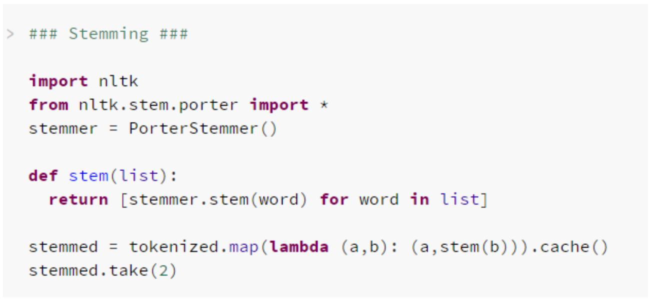
After this code was run, the data was tokenized and below is a snippet of how the data looked.



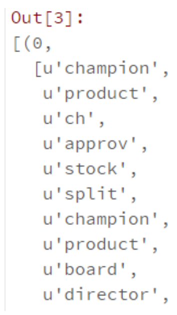
```
Out[2]:
[(0,
  [u'champion',
   u'products',
   u'ch',
   u'approves',
   u'stock',
   u'split',
   u'champion',
   u'products',
   u'board',
   u'directors',
   u'approved',
```

Step 3 – Stemming:

Stemming is the process of reducing inflected (or sometimes derived) words to their word stem, base or root form. [14] For stemming, the Porter stemming algorithm was used from the NLTK package. Below is the code for the stemming function along with a snippet of the output RDD after stemming.



```
> ### Stemming ###

  import nltk
  from nltk.stem.porter import *
  stemmer = PorterStemmer()

  def stem(list):
    return [stemmer.stem(word) for word in list]

  stemmed = tokenized.map(lambda (a,b): (a,stem(b))).cache()
  stemmed.take(2)
```

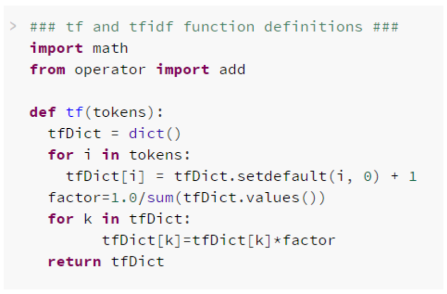


```
Out[3]:
[(0,
  [u'champion',
   u'product',
   u'ch',
   u'approv',
   u'stock',
   u'split',
   u'champion',
   u'product',
   u'board',
   u'director',
```

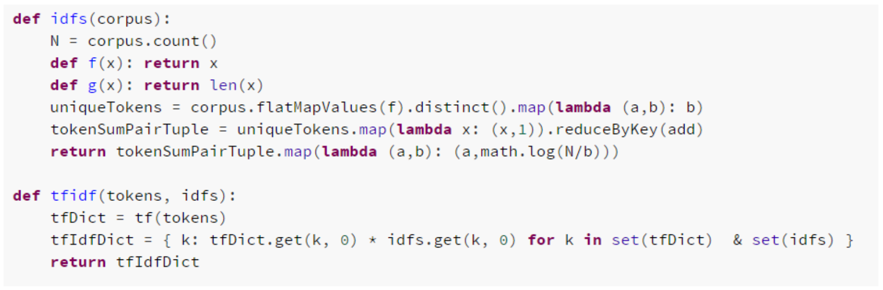
Step 4 - TF-IDF:

TF–IDF, short for term frequency–inverse document frequency, is a numerical statistic that is intended to reflect how important a word is to a document in a collection or corpus. Term frequency as the name suggests, is the frequency of occurrence of each word in a document. It is generally normalized to make the range uniform across documents. The inverse document frequency is a measure of how much information the word provides, that is, whether the term is common or rare across all documents. [15]

After all the data cleaning and processing was done, the TF-IDF values were computed. This was done using custom functions written in Python. The code snippets for the functions are below.



```
### tf and tfidf function definitions ###
import math
from operator import add

def tf(tokens):
  tfDict = dict()
  for i in tokens:
    tfDict[i] = tfDict.setdefault(i, 0) + 1
  factor=1.0/sum(tfDict.values())
  for k in tfDict:
      tfDict[k]=tfDict[k]*factor
  return tfDict
```



```
def idfs(corpus):
    N = corpus.count()
    def f(x): return x
    def g(x): return len(x)
    uniqueTokens = corpus.flatMapValues(f).distinct().map(lambda (a,b): b)
    tokenSumPairTuple = uniqueTokens.map(lambda x: (x,1)).reduceByKey(add)
    return tokenSumPairTuple.map(lambda (a,b): (a,math.log(N/b)))

def tfidf(tokens, idfs):
    tfDict = tf(tokens)
    tfIdfDict = { k: tfDict.get(k, 0) * idfs.get(k, 0) for k in set(tfDict)  & set(idfs) }
    return tfIdfDict
```

The tf function, given a list of token, returns a dictionary of all the unique terms and its normalized term frequency.

The idfs function takes a corpus as a parameter and calculates the idf values for the unique words. It returns another RDD which is of the format [(term1, idfvalue1), (term2, idfvalue2)…]. This is later reduced using .flatMap() to convert to a dictionary.

The tfidf function takes a list of tokens and the idf dictionary as parameters and calculates the TF-IDF scores and returns a dictionary of terms in the document with their TF-IDF values.

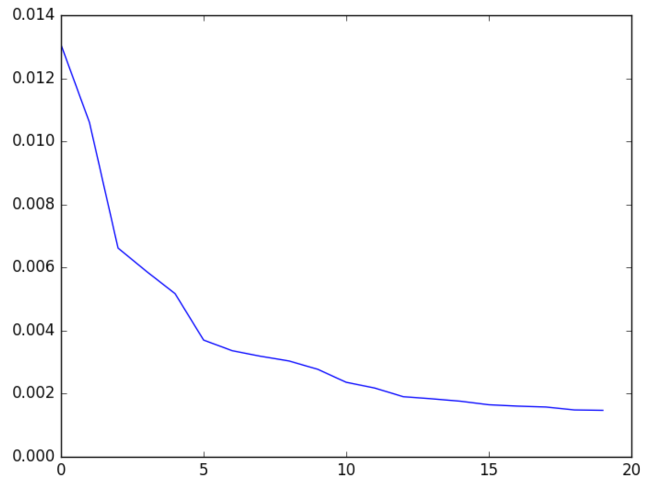
So, these functions were used on the datasets to obtain the TF-IDF values for each document and this was formatted to a 2D numpy array to create the TF-IDF document matrix.

Step 5 – PCA:

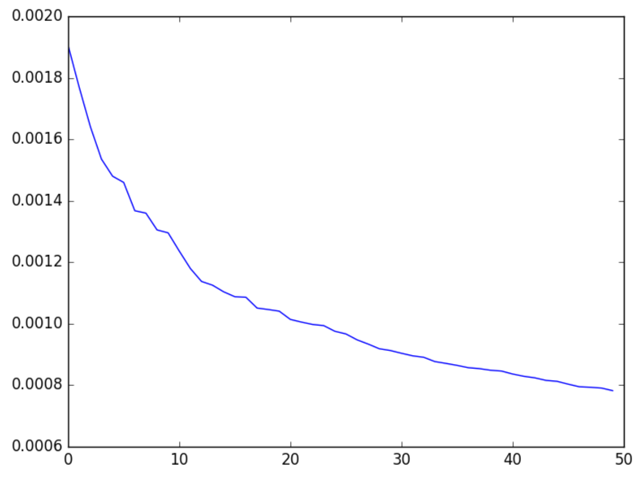
The Reuters data, after all the preprocessing had around 8000 features (columns) and the Ohsumed data had around 16000 features. Since the document matrices were very sparse, dimension reduction was required before sending the data to the clustering algorithms to get effective results.

Principal component analysis (PCA) is a statistical procedure that uses an orthogonal transformation to convert a set of observations of possibly correlated variables into a set of values of linearly uncorrelated variables called principal components [16]. PCA is a very commonly used technique for dimensionality reduction.

So, PCA was done using the PCA function in the scikit-learn library. In order to choose the optimum number of components, a scree plot was created using matplotlib. The scree plot for the Reuters data suggested 5 components (the number of factors corresponding to the maximum dip in the graph). It is shown below.



Similarly, a scree plot for the Ohsumed data was created and it is shown below.



The Ohsumed data was initially reduced to 12 components and clustering and result analysis was done. However, the results were not promising. Therefore, it was reduced to 6 components (the other big dip in the graph) and the results of that is discussed in the later sections.

3.3 Exploratory Data Analysis

This report will focus extensively on the Ohsumed dataset, with secondary focus on Reuters Dataset.

The distribution of both the datasets can be seen above under datasets.

a) Ohsumed

The process of EDA involves

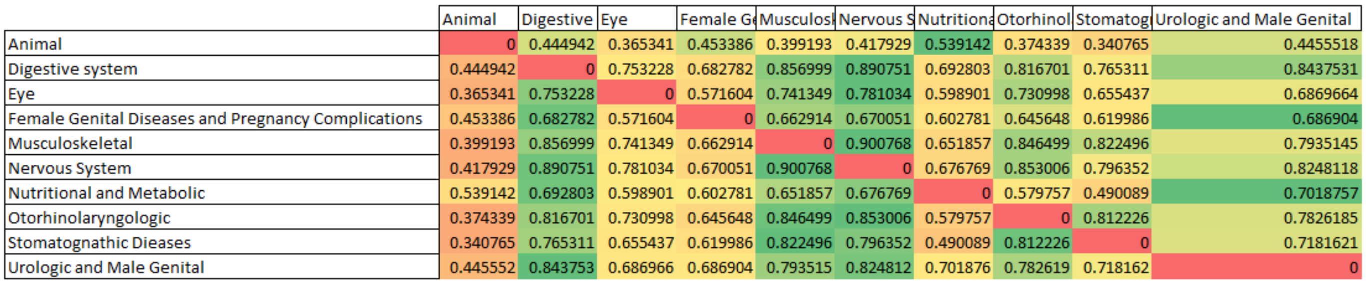
- Finding the Correlation Matrix for various categories of documents.
- Analysis on the term document matrix for the dataset.
- Explaining the need for TF-IDF and analysis on the TF-IDF matrix

Correlation Matrix

As mentioned in the description about Ohsumed dataset, it is a multi-Corpora and hence we can expect the data to have high degree of co-variance.

To create a correlation matrix, a normalized (based on count of categories) term document matrix was used. The TF-matrix initially had more than 16,000 features for the 3904 documents. The important features were filtered based on frequency of occurrence with features occurring in at least 2% of 3904 documents were retained. Finally, 525 features were used for EDA.

A cross-tab of categories and count of features in each category was taken. A correlation matrix was created on this data and it can be seen below.

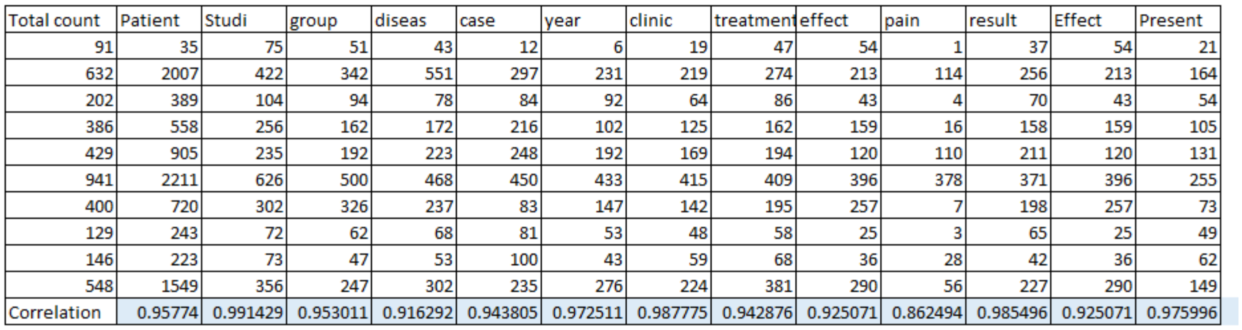


	Animal	Digestive	Eye	Female G	Musculosl	Nervous S	Nutritiona	Otorhinol	Stomatog	Urologic and Male Genital
Animal	0	0.444942	0.365341	0.453386	0.399193	0.417929	0.539142	0.374339	0.340765	0.4455518
Digestive system	0.444942	0	0.753228	0.682782	0.856999	0.890751	0.692803	0.816701	0.765311	0.8437531
Eye	0.365341	0.753228	0	0.571604	0.741349	0.781034	0.598901	0.730998	0.655437	0.6869664
Female Genital Diseases and Pregnancy Complications	0.453386	0.682782	0.571604	0	0.662914	0.670051	0.602781	0.645648	0.619986	0.686904
Musculoskeletal	0.399193	0.856999	0.741349	0.662914	0	0.900768	0.651857	0.846499	0.822496	0.7935145
Nervous System	0.417929	0.890751	0.781034	0.670051	0.900768	0	0.676769	0.853006	0.796352	0.8248118
Nutritional and Metabolic	0.539142	0.692803	0.598901	0.602781	0.651857	0.676769	0	0.579757	0.490089	0.7018757
Otorhinolaryngologic	0.374339	0.816701	0.730998	0.645648	0.846499	0.853006	0.579757	0	0.812226	0.7826185
Stomatognathic Dieases	0.340765	0.765311	0.655437	0.619986	0.822496	0.796352	0.490089	0.812226	0	0.7181621
Urologic and Male Genital	0.445552	0.843753	0.686966	0.686904	0.793515	0.824812	0.701876	0.782619	0.718162	0

In can be seen that all the categories are highly correlated to Musculoskeletal, digestive and Nervous system. This can be because those 3 contribute 60 % of the data. Additionally, Stomatognathic and Otorhinolaryngologic(ENT) as both are diseases around the ear, nose, larynx and throat regions. Also it can be noticed that male and female genitals are not highly correlated. This can be attributed to the fact that the dictionary for both differ although it talks about similar part of the body.

Term Document Matrix

The term document matrix was created for the 525 features. Despite a through pre-processing we found we found that there were a few words which would affect the model. These words were flagged and tracked throughout the analysis. These words followed the distribution of the categories and are given below.

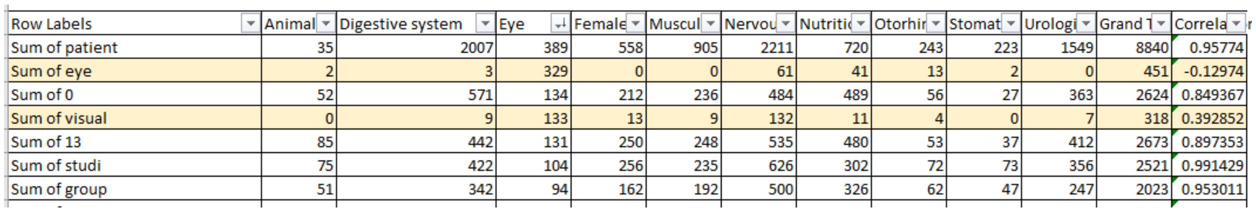


Total count	Patient	Studi	group	diseas	case	year	clinic	treatment	effect	pain	result	Effect	Present
91	35	75	51	43	12	6	19	47	54	1	37	54	21
632	2007	422	342	551	297	231	219	274	213	114	256	213	164
202	389	104	94	78	84	92	64	86	43	4	70	43	54
386	558	256	162	172	216	102	125	162	159	16	158	159	105
429	905	235	192	223	248	192	169	194	120	110	211	120	131
941	2211	626	500	468	450	433	415	409	396	378	371	396	255
400	720	302	326	237	83	147	142	195	257	7	198	257	73
129	243	72	62	68	81	53	48	58	25	3	65	25	49
146	223	73	47	53	100	43	59	68	36	28	42	36	62
548	1549	356	247	302	235	276	224	381	290	56	227	290	149
Correlation	0.95774	0.991429	0.953011	0.916292	0.943805	0.972511	0.987775	0.942876	0.925071	0.862494	0.985496	0.925071	0.975996

The table above shows the count of features/words for every category. The order of the categories is alphabetical. The correlation of each feature with total count of words in each category was identified.

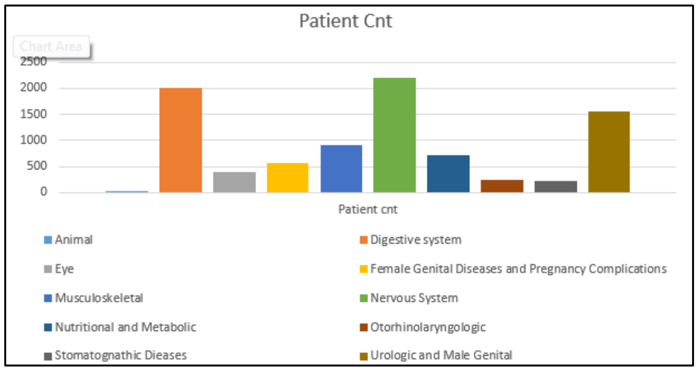
From the table we can notice that the above mentioned words have extremely high correlation with the distribution of data along the categories. These features/words trumped the effect of actual contributing words for each category.

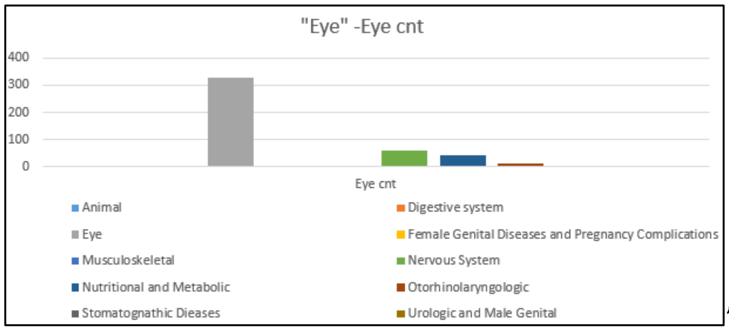
For example, in the category "Eye" it was noticed that the features "patient", "0", "study", "group" were top frequencies.



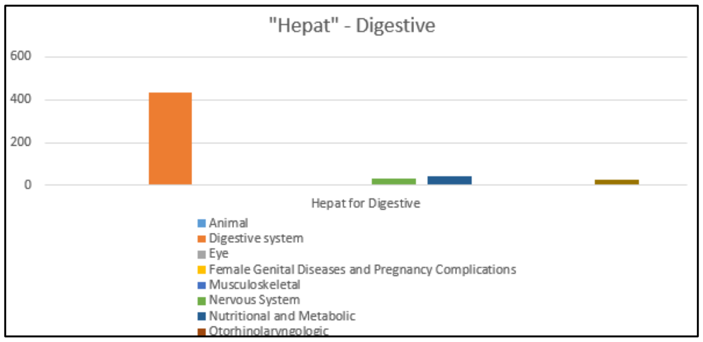
Row Labels	Animal	Digestive system	Eye	Female	Muscul	Nervou	Nutritic	Otorhir	Stomat	Urologi	Grand T	Correla
Sum of patient	35	2007	389	558	905	2211	720	243	223	1549	8840	0.95774
Sum of eye	2	3	329	0	0	61	41	13	2	0	451	-0.12974
Sum of 0	52	571	134	212	236	484	489	56	27	363	2624	0.849367
Sum of visual	0	9	133	13	9	132	11	4	0	7	318	0.392852
Sum of 13	85	442	131	250	248	535	480	53	37	412	2673	0.897353
Sum of studi	75	422	104	256	235	626	302	72	73	356	2521	0.991429
Sum of group	51	342	94	162	192	500	326	62	47	247	2023	0.953011

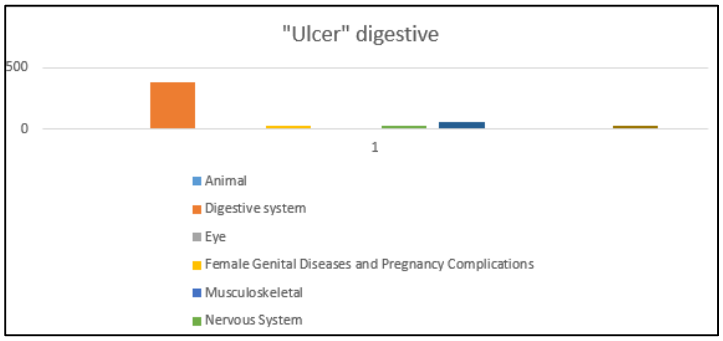
Furthermore, it can be noticed from the graph below the contributing features to each category. Each graph shows the count of that particular feature in all the categories

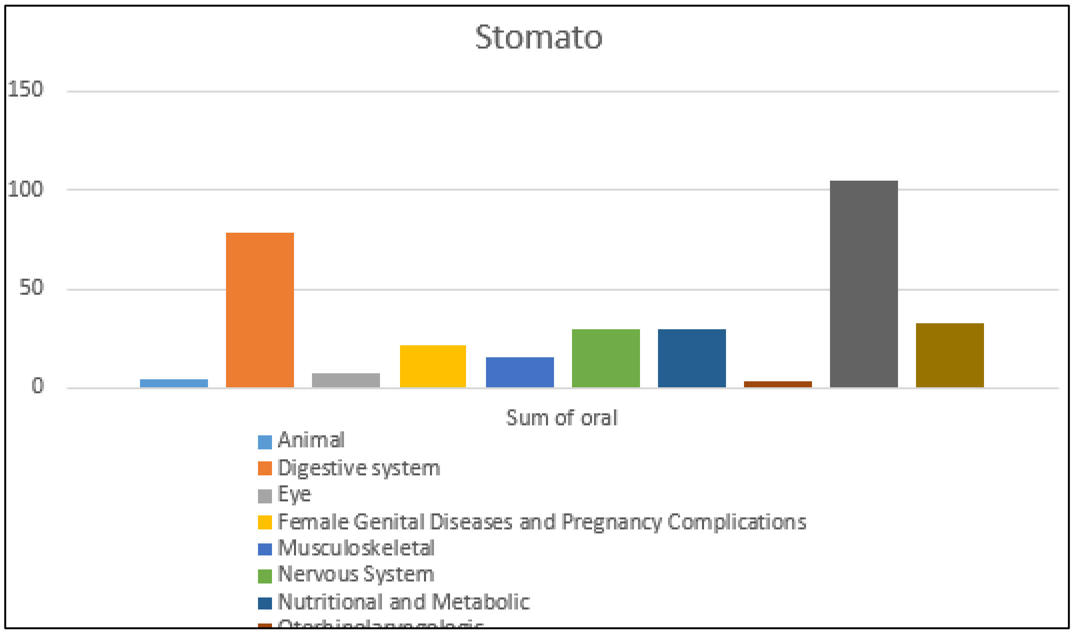


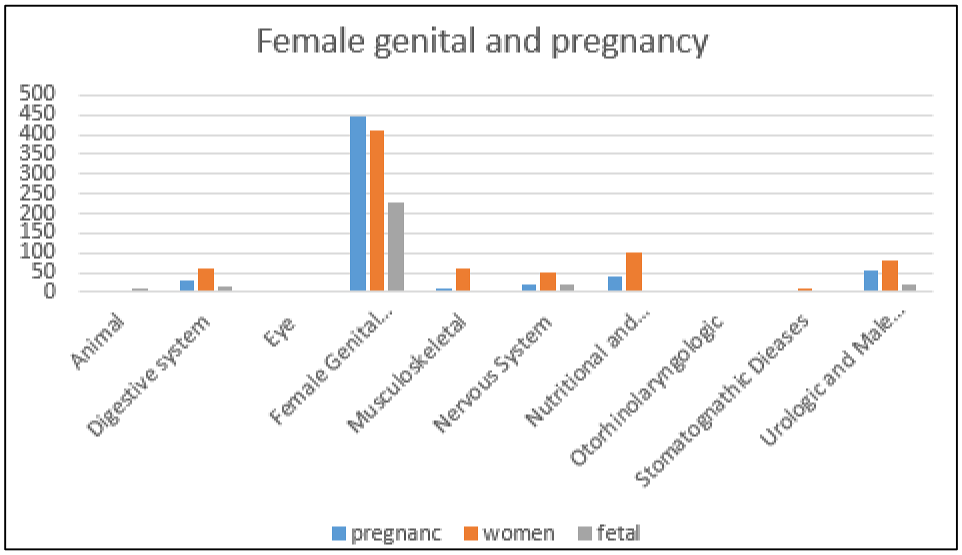


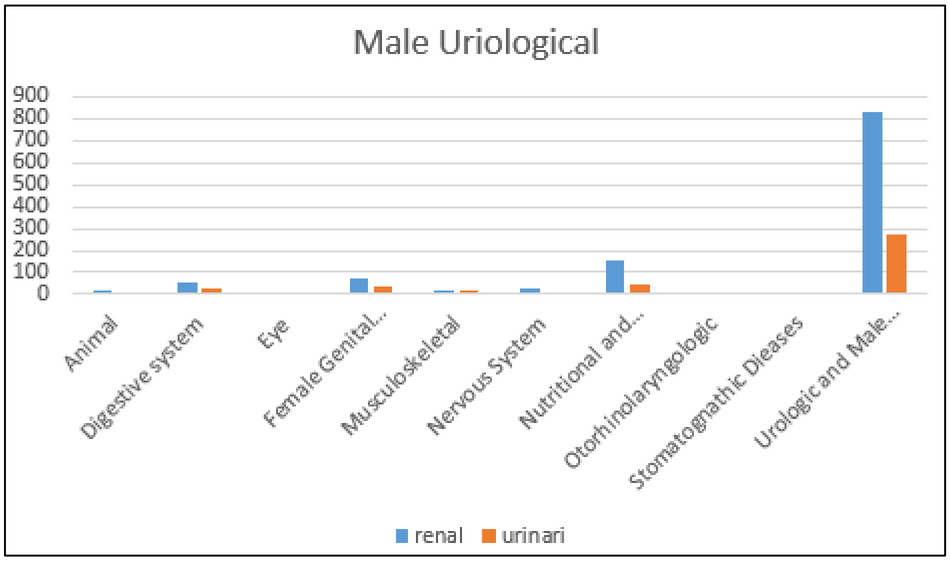
,











Above, are some of the important words in each document. Words like “Hepat”, “ Ulcer” exclusively explains the digestive system category. There various other words such as “Oral” which although predominantly exists in the Stomatognathic category, the word is present in other categories as well. Such words can contribute to multi class labelling.

To reduce the effect of words/features such as “patient” a TF-IDF matrix was created.

TF-IDF Matrix

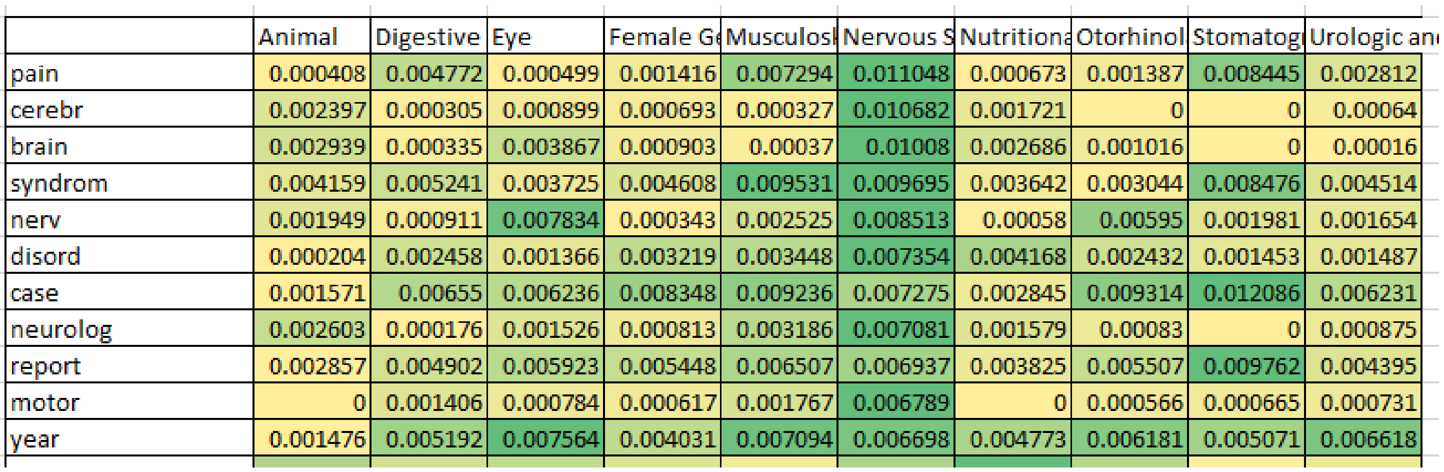
For the analysis of TF-IDF matrix, we picked nervous system as it contributes to approx 25% of the total no of documents in the analysis, hence it will be correlated to other topics.

We were successful is eliminating the effect of high-frequency, highly correlated to the data distribution, words such as "patient". The TF values became almost zero.

However other words such as "pain", "disorder", "syndrome" which intuitively seems to be words that will occur commonly in diseases related documents, did not occur as frequently as it was expected. Hence, this lead to such words being a contributing factor for each category.

Taking the example of nervous system, after calculating the TD-IDFs the average TF-IDFs for

"Nervous system"- All Feature combination can be seen below. We have shown the top 10 contributing features.



	Animal	Digestive	Eye	Female G	Musculosl	Nervous S	Nutritiona	Otorhinol	Stomatogı	Urologic an
pain	0.000408	0.004772	0.000499	0.001416	0.007294	0.011048	0.000673	0.001387	0.008445	0.002812
cerebr	0.002397	0.000305	0.000899	0.000693	0.000327	0.010682	0.001721	0	0	0.00064
brain	0.002939	0.000335	0.003867	0.000903	0.00037	0.01008	0.002686	0.001016	0	0.00016
syndrom	0.004159	0.005241	0.003725	0.004608	0.009531	0.009695	0.003642	0.003044	0.008476	0.004514
nerv	0.001949	0.000911	0.007834	0.000343	0.002525	0.008513	0.00058	0.00595	0.001981	0.001654
disord	0.000204	0.002458	0.001366	0.003219	0.003448	0.007354	0.004168	0.002432	0.001453	0.001487
case	0.001571	0.00655	0.006236	0.008348	0.009236	0.007275	0.002845	0.009314	0.012086	0.006231
neurolog	0.002603	0.000176	0.001526	0.000813	0.003186	0.007081	0.001579	0.00083	0	0.000875
report	0.002857	0.004902	0.005923	0.005448	0.006507	0.006937	0.003825	0.005507	0.009762	0.004395
motor	0	0.001406	0.000784	0.000617	0.001767	0.006789	0	0.000566	0.000665	0.000731
year	0.001476	0.005192	0.007564	0.004031	0.007094	0.006698	0.004773	0.006181	0.005071	0.006618

Such words (pain, syndrom, case, year) could have been removed, but we chose not to remove as such a process will not be feasible if it has to be scaled.

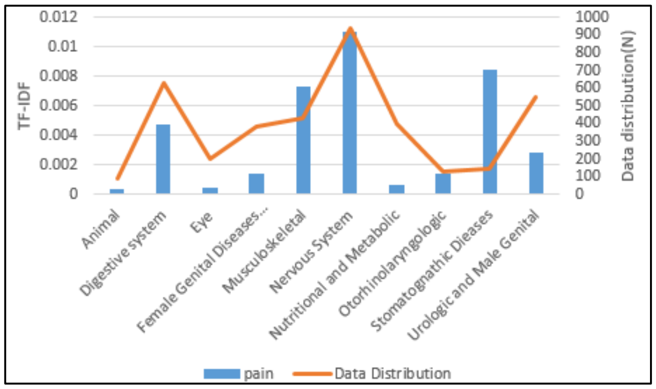
The color code shows the degree of values in each row(feature). It can be seen that words such as "nerv" have similar TF-IDF values for the category "Eye" and "Nervous system". Intuitively, it makes sense as various eye disease are related to nerves and hence the connection.

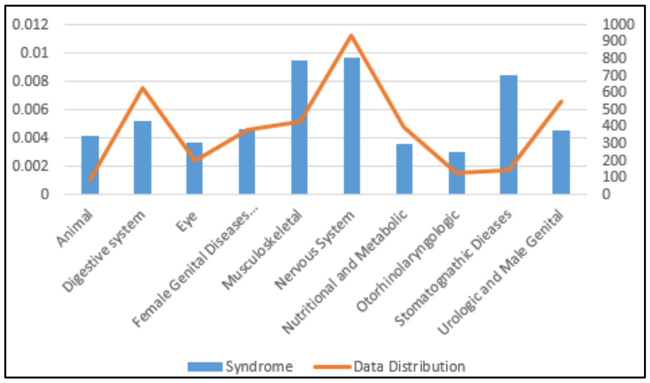
Various such cases also warrant the possibility of multi-class clustering on this data.

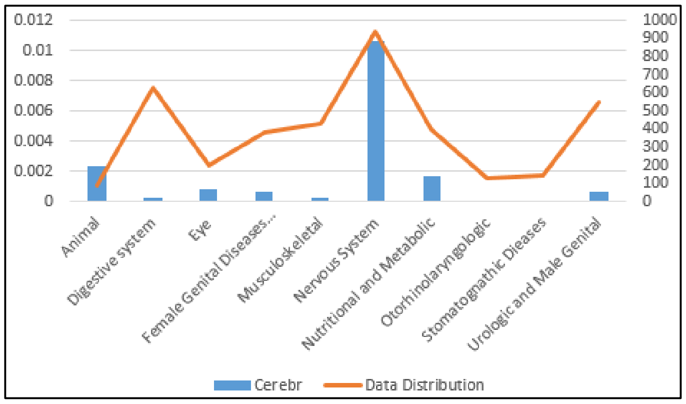
However, it can also be noticed that words such as "year" does not make intuitive sense to exist frequently in Eye and Nervous system categories. Such cases are purely document specific. We have not removed such cases as it is not practically scalable.

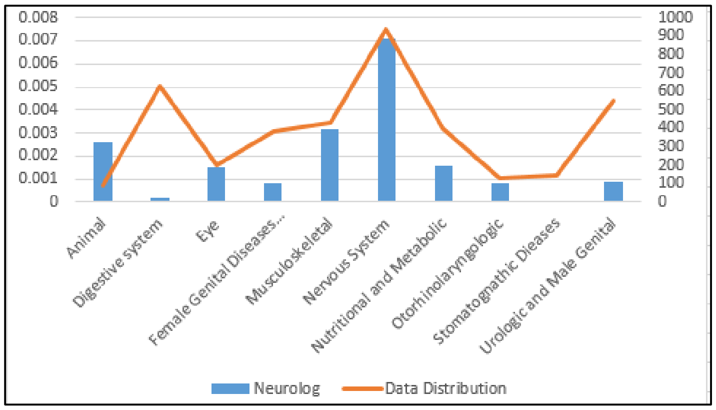
We are aware that such words can be a differentiating factor / noise while clustering. Ideally the input of such words to a clustering algorithm should be similar to that of the word patient.

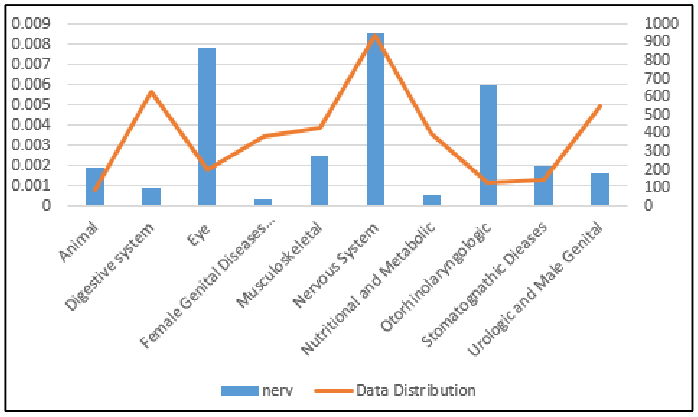
The graph below shows the **average** TF_IDF values compared to the data's distribution.

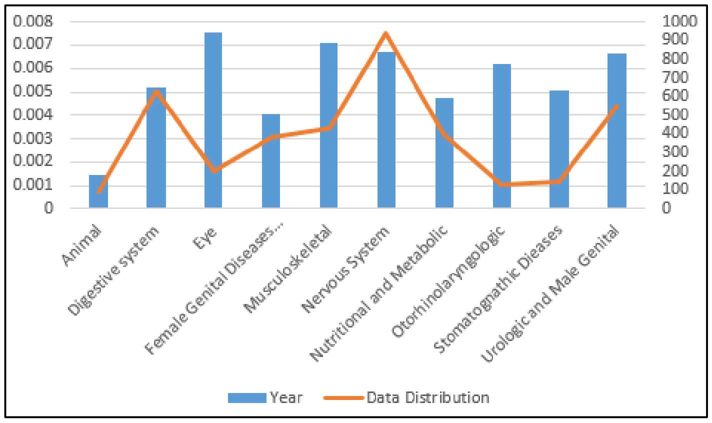












Words such as “pain”, “syndrome” despite being normalized follows the distribution of data. Other words such as “year”, “nerv” are not correlated to the distribution of data but have high correlation within categories.

From EDA, we infered that there will be a high possibility of Multi class labelling while clustering. We also noticed that there is possiblity of noise in the data i.e. words which might not make intutive sense to define its existence in the document, however becomes a differentiating factor while clustering. We decided not remove such words as it will not become scable to any type of data.

b) Reuters

The process of EDA also involved

- Finding the Correlation Matrix for various categories of documents.
- Analysis on the term document matrix for the dataset.
- Analysis on the TF-IDF matrix

However, to keep the report concise we have chosen not to share the results on the report.

4. Methodology

4.1 Text Data Preprocessing

Typical to any text mining pipeline, the data preprocessing step involved a number of steps including:

- Text cleaning (removing punctuations, whitespaces etc.)
- Tokenization
- Stop words removal
- Stemming
- TF-IDF calculation
- Dimension Reduction (Principal Component Analysis)

Details on each of these steps have been mentioned in the data preprocessing section 3.2 of the report earlier.

4.2 Fuzzy C-means Algorithm

For clustering the text corpus, we have used one of the modern clustering technique called Fuzzy C-means clustering. Simply said, it is a rule based clustering technique just like the K-means Clustering technique. *Additionally, it gives a membership matrix i.e a degree to which each document belongs to a cluster.* This is a probability value, with the sum of probability being equal to one for each document. Such a clustering technique can be very effective for multi-corpora problems where one document cannot be completely attributed to one single cluster.

In such a clustering technique, the tuning parameters become extremely important in determining a good cluster. Other than the general parameters such as error tolerance and # clusters, Fuzzy C means has another parameter called Fuzziness coefficient (m). This parameter controls the overlap of clusters. Higher the value, higher will be the degree of overlap, (m>1).

From [4], highlighting the important equations to write code an FCM algorithm is shown below.

FCM incorporates fuzzy membership values in its variance-based criterion as

$$J_m = \sum_{i=1}^{c} \sum_{j=1}^{n} u_{ij}^{m} \cdot d^2(x_j, v_i),$$

Where v_i is the center of cluster u_i.

The update equation for cluster centres and membership matrix can be seen below

$$v_i = \frac{\sum_{j=1}^{n} u_{ij} \,.\, x_j}{\sum_{j=1}^{n} u_{ij}},$$

and

$$u_{ij} = \left[\sum_{r=1}^{c} \left(\frac{d^2(x_j, v_i)}{d^2(x_j, v_r)} \right)^{1/(m-1)} \right]^{-1}.$$

This is an iterative process until the minimization function J_m hits a minimum point. The initial membership matrix is randomized so that the minimization function does not hit a local minimum.

The reason why this is different from K means is the way the error/ membership matrix is created. The value u_{ij} is the distance of a data point with one cluster divided by the distance of that point from other clusters. After taking an inverse, the membership matrix value will be high for cluster centers that are closer to the data point compared to the one that is farther. Hence a probability value is assigned for each data point against a cluster center.

We had interpreted the algorithm on our own and had implemented the complete algorithm in Python. This was later compared with an off the shelf Sckit-learn package for fuzzy c means clustering. The comparison of user defined algorithm with off the shelf algorithm was compared based on the metric called Fuzzy partition coefficient [5]. It should be between value 1/c and 1 where c=# of cluster.

The algorithm had 2 distance measure Cosine similarity and Euclidean distance. We continued to use Euclidean distance as the results were not very different for our sample datasets.

The algorithm on a high level is

1. Initializing the membership matrix
2. Calc. cluster centers
3. Calc. the minimization function.
4. Recalc. The membership matrix based on the centers created from step (2)
5. Go to Step and repeat until minimization point is reached at step (3) or max. iteration is reached.

Initialization of membership function

```
#random number generation for the initial membership matrix
e = np.random.random((c,l))
randtot=np.sum(e, axis=0)

#new Uik calc.
Uik=np.divide(e,randtot)
Uikm=(Uik)**m
```

Loop to compare the current minization value with the previous value

Inside the loop center centres calculation, membership function calculations are implemented. The while loop also has iteration b from which jmin is calculated for the while loop to function.

```
while (np.absolute(njmin-jmin)>tol and iterationcnt<req_iter):
    #################### Iteration a ######################
    #center calc.
    nvi=np.zeros(d)
    for i in range(dim):
        nvi[i]=np.sum(np.multiply(data[i],Uikm),axis=1)/np.sum(Uikm,axis=1)

    #new membership matrix calc.
    dataT=data.T
    nviT=nvi.T
    size=(c,l)
    nUik=np.zeros(size)
    for j in range(c):
        for i in range(l):
            xk_vi_dist=dist(dataT[i],nviT[j])
            denom=0
            for r in range(c):
                    xk_vr_dist=dist(dataT[i],nviT[r])
                    ratio=(xk_vi_dist/xk_vr_dist)**exp
                    denom=denom+ratio
    #new Uik is calc. here
            nUik[j,i]=1/denom

    #new Uikm is calc. here
    nUikm=(nUik)**m

    #minimization function calc.
    njm=np.zeros(size)
    for j in range(c):
        for i in range(len(dataT)):
            njm[j,i]=(dist(dataT[i],nviT[j]))**2

    njmin=np.sum(np.multiply(njm,nUikm))
```

These parts of the codes are used to obtain some key evaluation metrics such as FPC, cluster allocations and formatted membership matrices.

```
#this will have the final centers
newcenters=viT

#This will have the allocation of datapoints to centers
allocation=np.argmax(Uik.T,axis=1)

#This will have SSE
SSE=0
for i in range(l):
    SSE=SSE+dist(newcenters[allocation[i]],data.T[i])

#This will have SST
SST=0
for i in range(l):
    SST=SST+dist(data.T[i],np.mean(newcenters,0))

#This will have the membership matix U
mem_mat=Uik.T

#FPC : varies from (1/c to 1)
fpc=np.sum(Uik**2)/l

#FuPC (normalized to 0 to 1)
fpc_norm=(np.sum(Uik**2)/l-(1/c))/(1-(1/c))


print("The vectors below in the order are")
print("Allocation,centers,SSE,SST,mem_mat,fpc,fpc_norm,iterationcount")
return allocation+1,newcenters,SSE,SST,mem_mat,fpc,fpc_norm,iterationcnt
```

The FPC and membership values were compared between the off the shelf clustering and the user-defined fuzzy clustering technique. We noticed that the values changed by a meagre 1/100th of a decimal point depicting very high accuracy.

However, the efficiency of the code was not comparable to the off the shelf algorithm. For small data both methods were comparable. However, for Sparse vectors the user-defined algorithm where about 10 times slower.

4.3 K means Algorithm

For output analysis and comparison of performance of a fuzzy C means algorithm, an off the shelf K-means algorithm was used. We had calc. the inter cluster distance of data clustering using both algorithms.

In order to validate the fact that K means only assigns a data point to a single cluster and does not assign weights to the another possible cluster, this algorithm was compared with Fuzzy C means.

5. Evaluation

5.1 Strategy

The Fuzzy C -means clustering algorithm was used to cluster the 2 datasets (Ohsumed and Reuters). The result analysis is extensively done on Ohsumed dataset. From the EDA we identified that this is a Multi class problem. A Fuzzy C-means clustering algorithm can be advantageous over K-means as it gives the degree to with each the data points belong to each cluster.

Overview of the steps followed

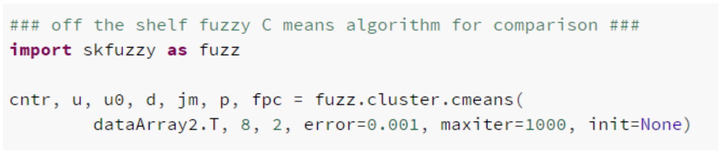
- Testing the algorithm: Test the user defined the Fuzzy-C clustering technique with various sample datasets and compare with an off-the-shelf technique
- Fuzzy- C with tuning: Run user defined Fuzzy-C means clustering on the dimension-reduced data using various tuning parameters
- Run an off the shelf K-means clustering on the dimension-reduced data
- Compare the results of the 2 clustering techniques and discuss the pros and cons of Fuzzy-C means technique over K means using confusion matrices, recall rate and cluster plots.

5.2 Results

Testing the algorithm

Since the algorithm of Fuzzy C means was created by us from the scratch it was tested with an off-the-shelf package. The results were compared for data with 1 dimension to upto 25 dimensions with 100 to 5000 records. The membership matrices of both techniques were compared and the difference in **Fuzzy partition coefficient** were calculated. The difference between the values for about 10 trials was less than 0.005% on an average. Hence, it proved that the algorithm created by us is robust.

Find below the off-the-shelf implementation of Fuzzy-C



```
### off the shelf fuzzy C means algorithm for comparison ###
import skfuzzy as fuzz

cntr, u, u0, d, jm, p, fpc = fuzz.cluster.cmeans(
        dataArray2.T, 8, 2, error=0.001, maxiter=1000, init=None)
```

Ohsumed

Since the data was sparse, we had to use a dimensionality reduction technique. We used PCA to reduce the dimensions. A scree plot was used to identify the “elbow point” for optimal number of dimensions. This data with 3904 documents and reduced features were clustered using Fuzzy-C.

Fuzzy- C with tuning:

The various tuning parameters used are

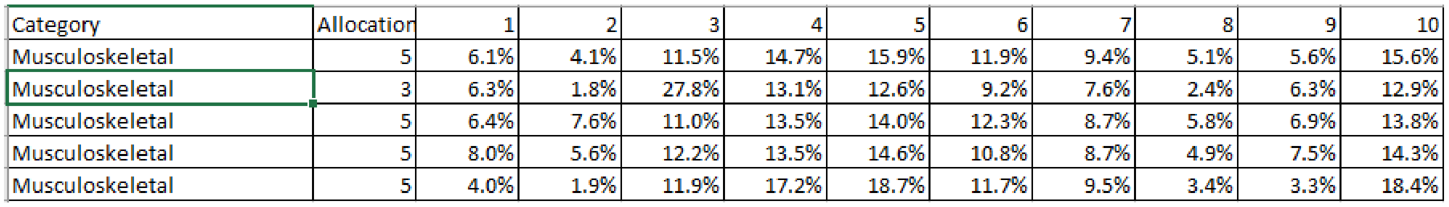
- Fuzzy coefficient (*m*)
- PCA elbow points (6 and 12)
- Error tolerance (*e*)
- # clusters

As mentioned earlier, increasing *m* will increase the fuzziness. We tested the same with value of *m*=5. All the data got clustered into a single cluster. This was because of the extreme value.

A standard value used for *m* in [5] is 2. This was tested with m=2 and #dimensions=12 after PCA. The confusion matrix can be seen below.

Also note that a data point was allocated to a cluster based on the degree of association being the highest amongst the clusters.

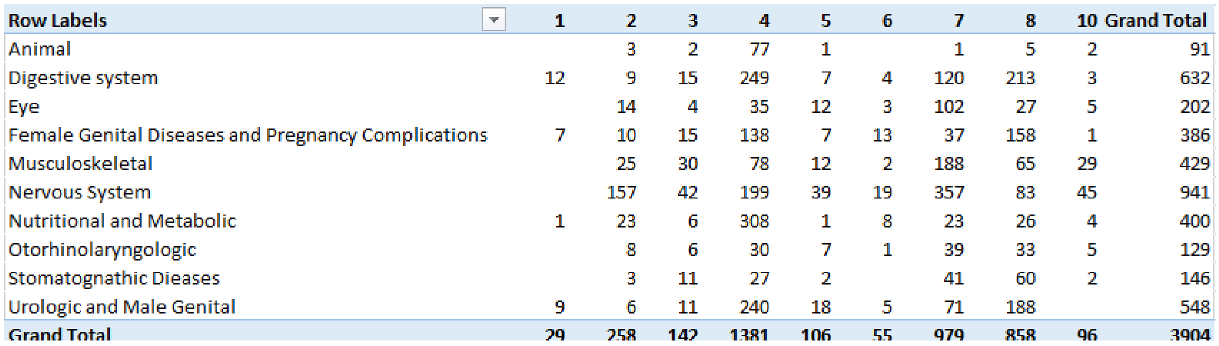
<u>Membership matrix and cluster allocation</u>



Category	Allocation	1	2	3	4	5	6	7	8	9	10
Musculoskeletal	5	6.1%	4.1%	11.5%	14.7%	15.9%	11.9%	9.4%	5.1%	5.6%	15.6%
Musculoskeletal	3	6.3%	1.8%	27.8%	13.1%	12.6%	9.2%	7.6%	2.4%	6.3%	12.9%
Musculoskeletal	5	6.4%	7.6%	11.0%	13.5%	14.0%	12.3%	8.7%	5.8%	6.9%	13.8%
Musculoskeletal	5	8.0%	5.6%	12.2%	13.5%	14.6%	10.8%	8.7%	4.9%	7.5%	14.3%
Musculoskeletal	5	4.0%	1.9%	11.9%	17.2%	18.7%	11.7%	9.5%	3.4%	3.3%	18.4%

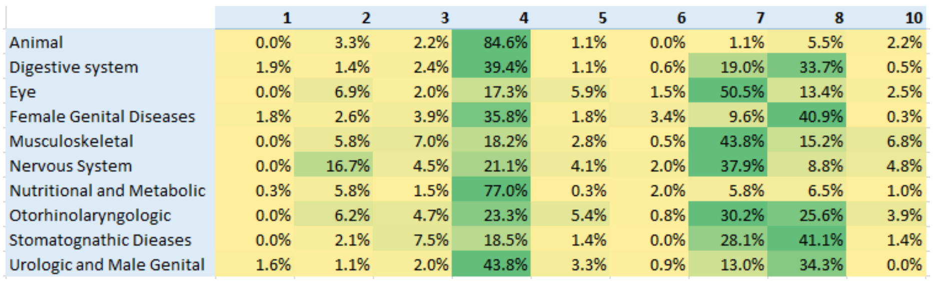
For example: row 2 has the highest degree of association of 27% to be the highest. Hence it is assigned to cluster 3.

<u>Confusion matrix</u>



Row Labels	1	2	3	4	5	6	7	8	10	Grand Total
Animal		3	2	77	1		1	5	2	91
Digestive system	12	9	15	249	7	4	120	213	3	632
Eye		14	4	35	12	3	102	27	5	202
Female Genital Diseases and Pregnancy Complications	7	10	15	138	7	13	37	158	1	386
Musculoskeletal		25	30	78	12	2	188	65	29	429
Nervous System		157	42	199	39	19	357	83	45	941
Nutritional and Metabolic	1	23	6	308	1	8	23	26	4	400
Otorhinolaryngologic		8	6	30	7	1	39	33	5	129
Stomatognathic Dieases		3	11	27	2		41	60	2	146
Urologic and Male Genital	9	6	11	240	18	5	71	188		548
Grand Total	**29**	**258**	**142**	**1381**	**106**	**55**	**979**	**858**	**96**	**3904**

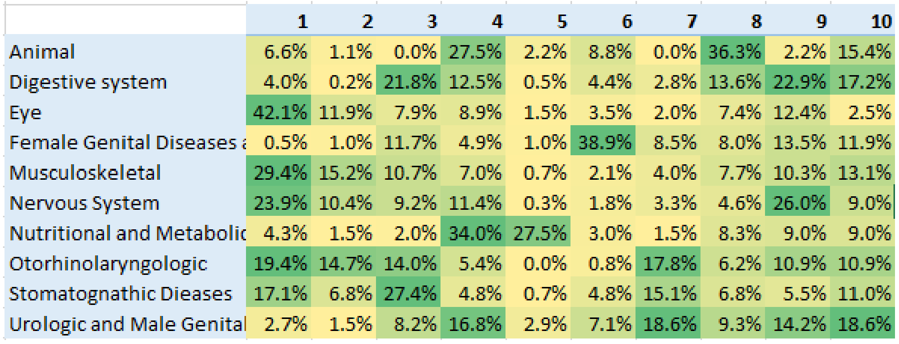
A percentage distribution of data can be seen below.



	1	2	3	4	5	6	7	8	10
Animal	0.0%	3.3%	2.2%	84.6%	1.1%	0.0%	1.1%	5.5%	2.2%
Digestive system	1.9%	1.4%	2.4%	39.4%	1.1%	0.6%	19.0%	33.7%	0.5%
Eye	0.0%	6.9%	2.0%	17.3%	5.9%	1.5%	50.5%	13.4%	2.5%
Female Genital Diseases	1.8%	2.6%	3.9%	35.8%	1.8%	3.4%	9.6%	40.9%	0.3%
Musculoskeletal	0.0%	5.8%	7.0%	18.2%	2.8%	0.5%	43.8%	15.2%	6.8%
Nervous System	0.0%	16.7%	4.5%	21.1%	4.1%	2.0%	37.9%	8.8%	4.8%
Nutritional and Metabolic	0.3%	5.8%	1.5%	77.0%	0.3%	2.0%	5.8%	6.5%	1.0%
Otorhinolaryngologic	0.0%	6.2%	4.7%	23.3%	5.4%	0.8%	30.2%	25.6%	3.9%
Stomatognathic Dieases	0.0%	2.1%	7.5%	18.5%	1.4%	0.0%	28.1%	41.1%	1.4%
Urologic and Male Genital	1.6%	1.1%	2.0%	43.8%	3.3%	0.9%	13.0%	34.3%	0.0%

It can be seen that a huge percentage of data gets clustered just in the clusters (4,7,8). Hence the dimensions were also reduced to 6. Since the fuzziness was high it was reduced to 1.5.

M=1.5 #dimensions(PCA)=6

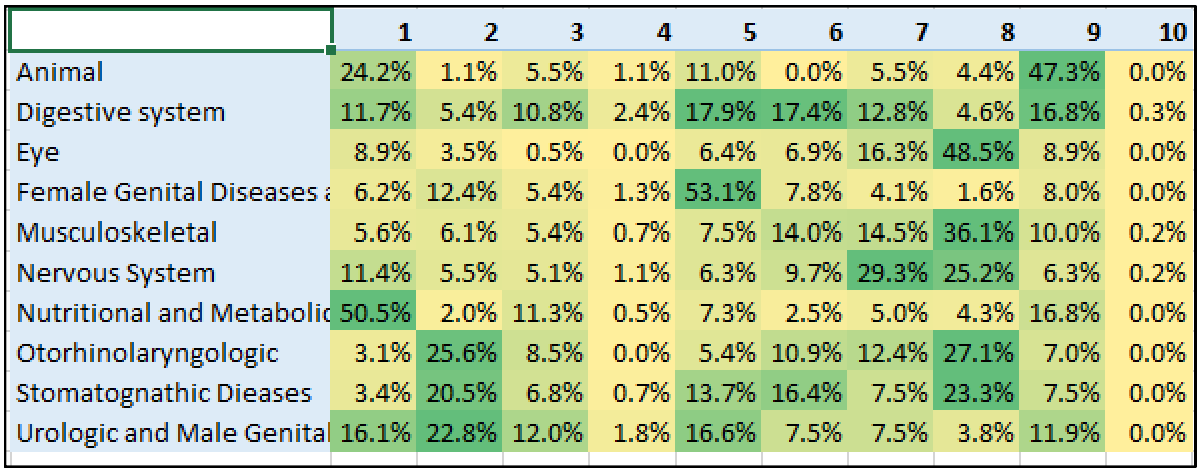


	1	2	3	4	5	6	7	8	9	10
Animal	6.6%	1.1%	0.0%	27.5%	2.2%	8.8%	0.0%	36.3%	2.2%	15.4%
Digestive system	4.0%	0.2%	21.8%	12.5%	0.5%	4.4%	2.8%	13.6%	22.9%	17.2%
Eye	42.1%	11.9%	7.9%	8.9%	1.5%	3.5%	2.0%	7.4%	12.4%	2.5%
Female Genital Diseases a	0.5%	1.0%	11.7%	4.9%	1.0%	38.9%	8.5%	8.0%	13.5%	11.9%
Musculoskeletal	29.4%	15.2%	10.7%	7.0%	0.7%	2.1%	4.0%	7.7%	10.3%	13.1%
Nervous System	23.9%	10.4%	9.2%	11.4%	0.3%	1.8%	3.3%	4.6%	26.0%	9.0%
Nutritional and Metabolic	4.3%	1.5%	2.0%	34.0%	27.5%	3.0%	1.5%	8.3%	9.0%	9.0%
Otorhinolaryngologic	19.4%	14.7%	14.0%	5.4%	0.0%	0.8%	17.8%	6.2%	10.9%	10.9%
Stomatognathic Dieases	17.1%	6.8%	27.4%	4.8%	0.7%	4.8%	15.1%	6.8%	5.5%	11.0%
Urologic and Male Genital	2.7%	1.5%	8.2%	16.8%	2.9%	7.1%	18.6%	9.3%	14.2%	18.6%

Since it is a multi-class problem a document category can belong to more than one cluster and one cluster can have more than one document class. In the case above it can be seen that a high percentage values on each document is distributed among clusters. For eg: Animal category has the top 2 percentage values in cluster 4 and 8. Hence the recall rate is 63.5%.

While looking at the membership matrix values it was noticed that the clusters allocation was done by a percentage value differences of less than 2%. This meant that error tolerance of clusters had to be reduced. It was reduced from 0.01% to 0.001%. This further improved the clustering.

<u>Final Fuzzy-C correlation matrix for parameters # *clusters =10 m*=1.5 *e*=0.0001% and # dimensions after PCA = 6</u>



	1	2	3	4	5	6	7	8	9	10
Animal	24.2%	1.1%	5.5%	1.1%	11.0%	0.0%	5.5%	4.4%	47.3%	0.0%
Digestive system	11.7%	5.4%	10.8%	2.4%	17.9%	17.4%	12.8%	4.6%	16.8%	0.3%
Eye	8.9%	3.5%	0.5%	0.0%	6.4%	6.9%	16.3%	48.5%	8.9%	0.0%
Female Genital Diseases a	6.2%	12.4%	5.4%	1.3%	53.1%	7.8%	4.1%	1.6%	8.0%	0.0%
Musculoskeletal	5.6%	6.1%	5.4%	0.7%	7.5%	14.0%	14.5%	36.1%	10.0%	0.2%
Nervous System	11.4%	5.5%	5.1%	1.1%	6.3%	9.7%	29.3%	25.2%	6.3%	0.2%
Nutritional and Metabolic	50.5%	2.0%	11.3%	0.5%	7.3%	2.5%	5.0%	4.3%	16.8%	0.0%
Otorhinolaryngologic	3.1%	25.6%	8.5%	0.0%	5.4%	10.9%	12.4%	27.1%	7.0%	0.0%
Stomatognathic Dieases	3.4%	20.5%	6.8%	0.7%	13.7%	16.4%	7.5%	23.3%	7.5%	0.0%
Urologic and Male Genital	16.1%	22.8%	12.0%	1.8%	16.6%	7.5%	7.5%	3.8%	11.9%	0.0%

The recall rates slightly improved compared to the one above. For eg: (recall rate for animal category is 73%).

To test the limit of the fuzzy coefficient we further reduced the value to 1.25. The membership matrix abruptly increases to a very high value of an average 0.7 probability when it assigned a data to a cluster. This was certainly an error since a cluster with a low inter cluster distance cannot have such high probability.

Hence we finalized the parameters *m*=1.5 *e*=0.0001% and # dimensions after PCA = 6 to be the one used for our comparisons.

The # clusters used in the dataset was constantly the # documents in the cluster i.e 10

Comparison of Final Fuzzy-C output with EDA

One cannot compare the TF-IDF values's contribution to its cluster allocation as the PCA changes the dimensions of the data.

However, we can notice that the 8th cluster has high percentage of values from the topics Eye, Musculoskeletal, Nervous, Otorhinolaryngologic and Stomatognathic Dieases. This relation between the categories based on various features were already discussed during the EDA and the above 5 categories were highly correlated. The words/features such as “nerv”, “year” were contributors of such a scenario.

For ease of comparison, let’s take the case Eye Vs Nervous system categories. The reason for the above can be simply correlated to the fact that words like "Nerv" have similar average TF-IDF values for categories "Eye" and Nervous Systems. Hence the end up getting clustered into the same cluster. Intuitively, it makes sense that various eye disease will have the word "Nerve" and its derived forms in it. However, a similar case exists for words like "Year" have similar TF-IDF values for the given categories. Such words, since randomly exists frequently only in these categories might bias the results.

Evaluation metrics for the clustering

<u>Fuzzy partition coefficient(FPC):</u> The FPC value is a measure of total error in the clustering technique. Typically, the value should be between 1/#clusters (0.1 in our case) to 1. A Value close to 1 indicates that the partition has happened

effectively. **The FPC value of our case was 0.21**. **This value was compared with result from the of-the-shelf clustering technique and it matched our result**. This approved the fact that there the clusters are not spread wide apart.

Recall rate calculation: Out of the standard evaluation measures, such as F-measure, recall rate, precision, recall rate was chosen as the measure of comparison as it gives the true positive rate which is the most important metric of evaluation. The other measures are usually used for single class problems.

	Precision rate
Animal	71.4%
Digestive system	35.3%
Eye	64.9%
Female Genital Diseases a	65.5%
Musculoskeletal	50.6%
Nervous System	54.5%
Nutritional and Metabolic	67.3%
Otorhinolaryngologic	52.7%
Stomatognathic Dieases	43.8%
Urologic and Male Genital	39.4%

Comparison with K-means results

The confusion matrix of K-means results is given below

K means confustion matrix	1	2	3	4	5	6	7	8	9	10
Animal	31.9%	0.0%	2.2%	42.9%	0.0%	0.0%	9.9%	1.1%	0.0%	12.1%
Digestive system	21.8%	0.2%	0.3%	17.7%	0.0%	21.7%	28.2%	0.2%	2.5%	7.4%
Eye	12.9%	0.5%	1.0%	7.4%	0.0%	9.9%	21.8%	41.1%	1.5%	4.0%
Female Genital Diseases and Pregnancy Complications	11.9%	0.3%	1.0%	9.6%	0.0%	10.1%	13.7%	0.5%	8.8%	44.0%
Musculoskeletal	11.2%	0.0%	0.7%	9.8%	2.8%	16.1%	29.1%	23.1%	3.7%	3.5%
Nervous System	18.0%	0.1%	0.3%	6.4%	1.0%	10.8%	41.6%	15.7%	3.3%	2.9%
Nutritional and Metabolic	43.0%	0.0%	25.8%	12.5%	0.0%	4.8%	7.5%	2.3%	1.0%	3.3%
Otorhinolaryngologic	12.4%	0.0%	0.0%	7.8%	0.0%	20.2%	24.8%	15.5%	18.6%	0.8%
Stomatognathic Dieases	8.9%	0.0%	0.7%	8.2%	0.0%	25.3%	20.5%	13.0%	15.1%	8.2%
Urologic and Male Genital	24.3%	0.0%	2.2%	15.1%	0.0%	9.7%	18.1%	2.0%	18.6%	10.0%

The phenomenon of eye, musculoskeletal, nervous system, Otorhinolaryngologic, stomato. diseases being correlated, continues to exist even in K means clustering.

K means	Precision rate
Animal	74.7%
Digestive system	50.0%
Eye	62.9%
Female Genital Diseases ar	57.8%
Musculoskeletal	52.2%
Nervous System	59.5%
Nutritional and Metabolic	68.8%
Otorhinolaryngologic	45.0%
Stomatognathic Dieases	45.9%
Urologic and Male Genital	42.3%

The precision rates did not vary much for both Fuzzy C clustering and K means clustering as the average precision rate was 55% in both cases.

Inter cluster distance: The inter cluster distance was measured by finding the distance between the centers. The results of K- means and Fuzzy-C were compared.

K means	1	2	3	4	5	6	7	8	9	10
0	0	1.102021	0.14503	0.053543	0.467922	0.078926	0.046116	0.096682	0.135587	0.091474
1	1.102021	0	1.110802	1.096436	1.198828	1.085981	1.102007	1.105797	1.099007	1.105843
2	0.14503	1.110802	0	0.159412	0.49168	0.179585	0.168665	0.18771	0.204047	0.185684
3	0.053543	1.096436	0.159412	0	0.465065	0.091493	0.084042	0.123557	0.109874	0.113121
4	0.467922	1.198828	0.49168	0.465065	0	0.45391	0.467162	0.471998	0.485164	0.473733
5	0.078926	1.085981	0.179585	0.091493	0.45391	0	0.049886	0.076557	0.09392	0.086776
6	0.046116	1.102007	0.168665	0.084042	0.467162	0.049886	0	0.064906	0.127948	0.083238
7	0.096682	1.105797	0.18771	0.123557	0.471998	0.076557	0.064906	0	0.134695	0.142892
8	0.135587	1.099007	0.204047	0.109874	0.485164	0.09392	0.127948	0.134695	0	0.148976
9	0.091474	1.105843	0.185684	0.113121	0.473733	0.086776	0.083238	0.142892	0.148976	0

Average intercluster distance =0.378

Fuzzy C	1	2	3	4	5	6	7	8	9	10
0	0	0.1072	0.033155	0.049698	0.048785	0.058755	0.043741	0.078723	0.031854	0.048667
1	0.1072	0	0.086474	0.083964	0.086498	0.076621	0.095927	0.09496	0.090468	0.086029
2	0.033155	0.086474	0	0.017411	0.017129	0.027203	0.023596	0.059103	0.043446	0.016841
3	0.049698	0.083964	0.017411	0	0.00399	0.013706	0.021917	0.051373	0.059645	0.003237
4	0.048785	0.086498	0.017129	0.00399	0	0.017482	0.022963	0.054419	0.059611	8.13E-04
5	0.058755	0.076621	0.027203	0.013706	0.017482	0	0.025513	0.042049	0.065995	0.016834
6	0.043741	0.095927	0.023596	0.021917	0.022963	0.025513	0	0.04033	0.061748	0.022577
7	0.078723	0.09496	0.059103	0.051373	0.054419	0.042049	0.04033	0	0.091415	0.05384
8	0.031854	0.090468	0.043446	0.059645	0.059611	0.065995	0.061748	0.091415	0	0.059353
9	0.048667	0.086029	0.016841	0.003237	8.13E-04	0.016834	0.022577	0.05384	0.059353	0

Average intercluster distance =0.05

The lower inter cluster distance can be attributed to the fact that Fuzzy-C naturally aims at overlap of clusters. Although from this metric attributes to the fact that K-means is better than Fuzzy-C. K means does not have the scope to assign and justify multi class labels like Fuzzy-C can do. This can be seen below

Evaluation using membership matrix

In order to aggregate and analyze the membership matrix results, we have taken average values of membership matrix for each category, every time the category was predicted into the cluster.

Musculoskeletal	**8.0%**	**5.4%**	**9.3%**	**9.6%**	**9.2%**	**10.7%**	**12.8%**	**17.9%**	**7.8%**	**9.3%**
6	5.9%	7.1%	9.6%	12.2%	11.2%	16.9%	9.8%	9.9%	6.0%	11.4%
7	6.0%	1.6%	9.4%	10.2%	9.8%	10.0%	28.7%	11.0%	3.4%	9.9%
8	4.5%	4.2%	6.6%	8.0%	7.5%	10.5%	11.5%	36.0%	3.6%	7.6%

For example: The top 3 clusters for Musculoskeletal disease are clusters 6,7 and 8. The number 28.7% for cluster 7 means that every time cluster 7 is predicted as the allocated cluster, the average membership value for such cases is 28.7%. Similarly, the membership value every time cluster 8 was predicted is 36%, cluster 6 is 16.9%.

One can notice that when cluster 8 was predicted, cluster 6 and cluster 7 has second highest membership values. A similar behavior can be noticed for cluster 7. This property validates the fact that a category of documents can belong to more than one cluster and vice versa.

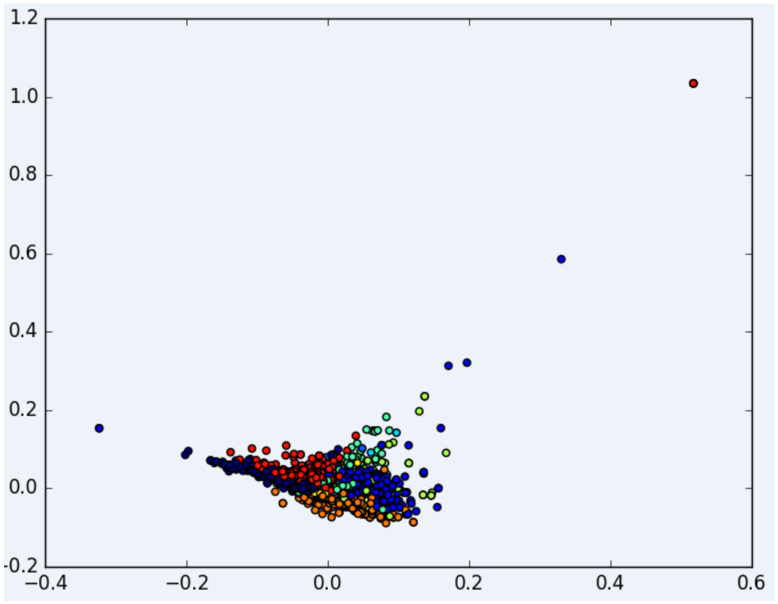
Another example with nutritional category can be seen below.

Nutritional and Metabolic	**18.9%**	**4.4%**	**12.1%**	**9.1%**	**9.3%**	**7.9%**	**10.0%**	**6.1%**	**12.9%**	**9.3%**
1	26.4%	4.0%	10.9%	8.0%	8.1%	6.9%	9.2%	5.4%	12.8%	8.2%
3	10.6%	2.3%	23.1%	11.4%	11.8%	8.0%	10.5%	3.4%	7.2%	11.8%
9	16.3%	5.6%	10.3%	7.5%	7.5%	6.8%	7.5%	4.9%	26.1%	7.5%

5.3 Discussion

From all the above analysis it can be concluded that the Ohsumed dataset is a multi-class corpora and the clusters allocated to these categories are multi labelled. The preprocessing has been done extensively on the data to remove, stop words and use effective algorithms such as a porter algorithm to stem the words. An extensive EDA was done to identify the features that can contribute to the clustering. Words like "Patient" which occurs frequently were removed by TF-IDF however words such as "pain" and "diseas" continued to exist as they were not as frequent. These words ended up being contributing factors to the clustering. Since we did not find a generalized metric to remove such words we continued to have them in the data. The membership matrix aligns with the results with the probability values being smaller values which means that the algorithm is not completely sure if the document belongs to a particular cluster.

Find below the Clusters after a PCA was done on the 6-dimension data to reduce it to 2 dimensions for the plot.

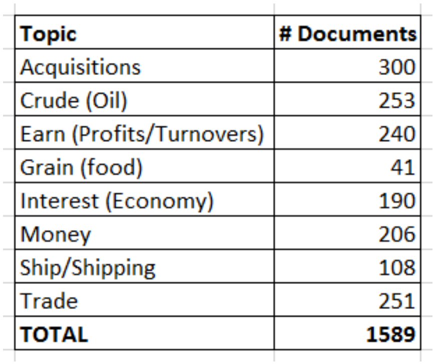


<u>Next steps</u>

The words similar to "pain", "diseas", "syndrom" can be added to the list of stop words as stop words while clustering "Medical conditions / diseases". The data and analysis can also be done using cosine similarity measure.

<u>Reuters</u>

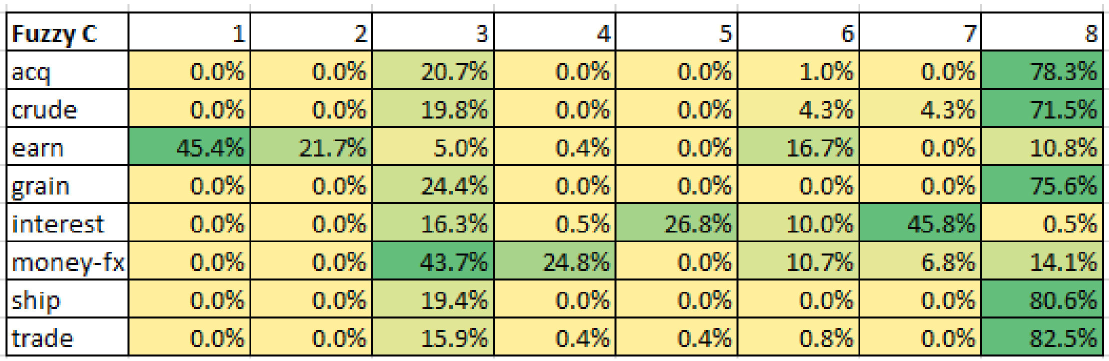
The output analysis was not done as extensively as it was done for Ohsumed dataset. At a high level, after EDA we noticed that there were features having similar TF-IDF values for the topics Acquisitions, Shipping and trade.



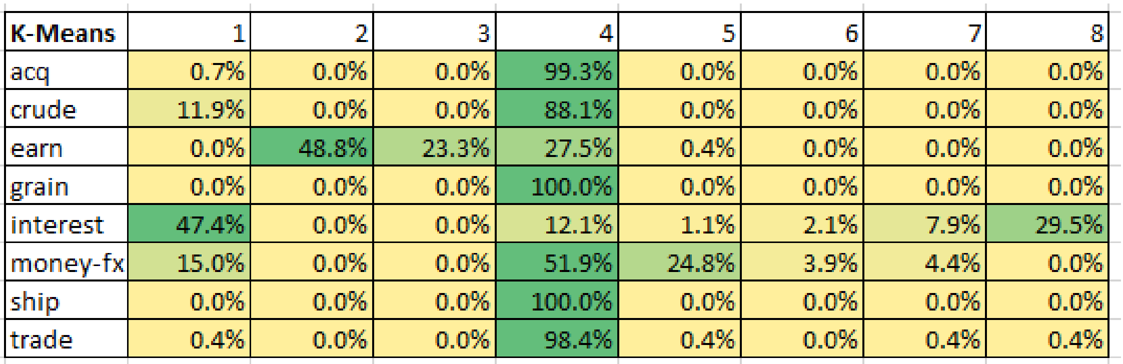
Topic	**# Documents**
Acquisitions	300
Crude (Oil)	253
Earn (Profits/Turnovers)	240
Grain (food)	41
Interest (Economy)	190
Money	206
Ship/Shipping	108
Trade	251
TOTAL	**1589**

The optimal tuning parameters were identified as *m*=2 #dimensions=5 error tolerance=0.01% #clusters=8

Comparison of K means with Fuzzy -C means



Fuzzy C	1	2	3	4	5	6	7	8
acq	0.0%	0.0%	20.7%	0.0%	0.0%	1.0%	0.0%	78.3%
crude	0.0%	0.0%	19.8%	0.0%	0.0%	4.3%	4.3%	71.5%
earn	45.4%	21.7%	5.0%	0.4%	0.0%	16.7%	0.0%	10.8%
grain	0.0%	0.0%	24.4%	0.0%	0.0%	0.0%	0.0%	75.6%
interest	0.0%	0.0%	16.3%	0.5%	26.8%	10.0%	45.8%	0.5%
money-fx	0.0%	0.0%	43.7%	24.8%	0.0%	10.7%	6.8%	14.1%
ship	0.0%	0.0%	19.4%	0.0%	0.0%	0.0%	0.0%	80.6%
trade	0.0%	0.0%	15.9%	0.4%	0.4%	0.8%	0.0%	82.5%



K-Means	1	2	3	4	5	6	7	8
acq	0.7%	0.0%	0.0%	99.3%	0.0%	0.0%	0.0%	0.0%
crude	11.9%	0.0%	0.0%	88.1%	0.0%	0.0%	0.0%	0.0%
earn	0.0%	48.8%	23.3%	27.5%	0.4%	0.0%	0.0%	0.0%
grain	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%
interest	47.4%	0.0%	0.0%	12.1%	1.1%	2.1%	7.9%	29.5%
money-fx	15.0%	0.0%	0.0%	51.9%	24.8%	3.9%	4.4%	0.0%
ship	0.0%	0.0%	0.0%	100.0%	0.0%	0.0%	0.0%	0.0%
trade	0.4%	0.0%	0.0%	98.4%	0.4%	0.0%	0.4%	0.4%

In both clustering algorithms every category is concentrated on just one cluster. The inter cluster distance for K means was 0.84 and Fuzzy-C means was 0.33.

A further deep dive with appropriate tuning parameters can improve the results.

6. Bibliography

1. Huang, Anna. "Similarity measures for text document clustering." Proceedings of the sixth new zealand computer science research student conference (NZCSRSC2008), Christchurch, New Zealand. 2008.
2. Steinbach, Michael, George Karypis, and Vipin Kumar. "A comparison of document clustering techniques." KDD workshop on text mining. Vol. 400. No. 1. 2000.
3. Renukadevi, D., and S. Sumathi. "Term based similarity measure for text classification and clustering using fuzzy C-means algorithm." Int. J. Sci., Eng. Technol. Res.(IJSETR) 3.4 (2014): 1093-1097.
4. Moertini, D., Veronica. "Introduction to Five Data Clustering Algorithms" INTEGRAL, Vol. 7, No. 2, Oktober 2002
5. Mohammad, Rawashdeh and Anca, Ralescu " Fuzzy Cluster Validity with Generalized Silhouettes" School of Computing Sciences and Informatics, University of Cincinnati
6. http://spark.apache.org/docs/0.9.0/python-programming-guide.html
7. https://databricks.com/
8. http://www.daviddlewis.com/resources/testcollections/reuters21578/
9. http://ana.cachopo.org/datasets-for-single-label-text-categorization
10. http://davis.wpi.edu/xmdv/datasets/ohsumed.html
11. http://disi.unitn.it/moschitti/corpora.htm
12. https://en.wikipedia.org/wiki/Tokenization_(lexical_analysis)

13. https://sites.google.com/site/kevinbouge/stopwords-lists
14. https://en.wikipedia.org/wiki/Stemming
15. https://en.wikipedia.org/wiki/Tf%E2%80%93idf
16. https://en.wikipedia.org/wiki/Principal_component_analysis

Individual Section

Arjhun Hariharan

Contributions

My major responsibility in this project was in the dataset collection, cleaning and preprocessing step. It started right from finding the source to download the dataset, to presenting the data in a format that the clustering algorithm accepts.

Since I was quite new to PySpark, this was a pretty challenging step as I had to refer and learn to write efficient MapReduce code apart from solving our specific problem. I wanted use this opportunity to learn PySpark (MapReduce code) as much as possible. So, for most of the preprocessing steps, I wrote custom functions and didn't use existing packages and functions. This includes data cleaning, tokenization, stop words removal and tf-idf calculations, as it can be seen from the code and the snippets provided. However, for stemming and PCA, existing packages were used.

Before starting to code however, I had to do some preliminary exploratory data analysis to see the distributions of documents across each categories and the average size of each documents. This was very important given our computation restrictions. So, we decided initially that we will only use a small subset of the entire corpus. Moreover, I also noticed that the distributions of the documents in each category were extremely different. While some categories had only a few tens of documents, others had thousands of documents. So I made sure the subset we chose had categories which had comparable number of documents.

As for the report, I was responsible from section 1 till the end of section 3.2 and section 6. Apart from that, I wrote the README file for the code and cleaned up the code to make it consumable.

Setbacks

Fortunately, there was not a single big show stopper that came my way during the course of this project.

However, one setback for us was regarding saving and opening the dataset. We decided we will use the Databricks platform for writing and executing our programs as their free community edition gave us a free 6 GB cluster. We decided to use Amazon S3's free trial for data storage and retrieval. However, within a few days of starting the data preprocessing, I found out that we had crossed our free tier limit. So I had to use Databrick's own file system as the place for the dataset to reside and used Amazon S3 only for saving the results.

Apart from this, the other ones were around data formatting and figuring out efficient ways to convert between different datatypes to have the data compatible for the clustering algorithms.

I found the overall process challenging as I had to deal with project related complications apart from learning PySpark in parallel.

Lessons Learnt

The entire project was a learning in many aspects. Learning a new framework, learning a new clustering technique, understanding the text mining and information retrieval processes etc.

The biggest takeaway for me was that despite so many challenges, time constraints and other issues, I enjoyed the work. This is proof that I am really interested in this space and that being a data scientist will make me happy! ☺

Varun Mohan Kumar

Contributions

My contribution to this project was to identify the importance of Fuzzy-C means clustering algorithm in text mining. I had learnt the algorithm understood the importance of such a technique in text mining.

I had written a custom function for Fuzzy-C clustering in Pyspark. This step was challenging as I was learning python for the first time. The 2 biggest challenges; one was to understand the Fuzzy-C means algorithm as it is complex to implement and not as easy as K means clustering algorithm. The second challenge was to understand the syntax of python as I was new to the software. It was also very difficult to identify errors in the algorithm as there were too many matrices to keep track of.

I had done an extensive EDA on Ohsumed dataset and a high level EDA on Reuters dataset. The no of features was higher than usual data analyses we had done earlier. To consolidate the chunk of data was exciting. Visualization the data with color codes helped my case.

I had worked on the evaluation of results from Ohsumed and Reuters. It was challenging to come out of the frame of mind of a single class clustering to a multi-class clustering. Initially it seemed as though algorithm was not effective. But as we know the data never lies. A further deep dive helps us understand the data better.

In the report the sections I have covered are from Section 3.3 until end of section 5.

Setbacks

The biggest setback in my case was not knowing what not knowing what to do with medical stop words such as “diseas”, “pain” etc. Since they did not get removed by TF-IDF and became contributing factor it became unexplainable factors/ error. We could have added these words to a TF-IDF list but it made the process not scalable. That is the only solution we found for such cases.

Lessons Learnt

The entire project was a learning in many aspects. Learning a new framework, learning a new clustering technique, understanding the text mining and information retrieval processes etc.

The important takeaway for me is that one has to be tool agnostic. The skill to analyse the data is much more important than writing a code. It was great explore one the difficult problems and not give up until the very end to give fruitful results.

One cannot become a skilled sailor by simply sailing on smooth seas ☺ !!!!!