[DTI5125 [EG] Data Science Applications
Group 4, Assignment 2, Text Clustering]

[AbdelMageed Ahmed AbdelMageed Hassan]

[Ahmed Shehata Mahmoud AboMooustafa]

[Sarah Hossam AbdelHameed Elmowafy]

[Mohamed Sayed AbdelWahab Hussien]

Abstract

Text clustering is the task of grouping a set of unlabeled texts in such a way that texts in the same cluster are more similar to each other than to those in other clusters. Having a number of documents extracted from five Gutenberg books of different genres, this report discusses the methodologies and models with different transformation techniques that have been applied to these documents to see if the natural clusters could be extracted out of these data one more time.

Introduction

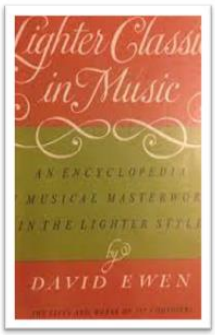
Text is not just some scattered letters, its value is much greater than that, however it's not an easy job to excavate this value or in other words to extract insights from it due to its unstructured nature. The objective here is to prepare a set of random data and apply clustering methods on it and watch if we could get clusters that are close to the original books; analyze the pros and cons of algorithms and generate and communicate the insights so, the implementation is needed to apply these methods and transform text then evaluate each one. the best language to be used in such problems is python with its vast libraries.

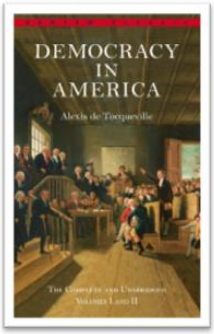
Dataset

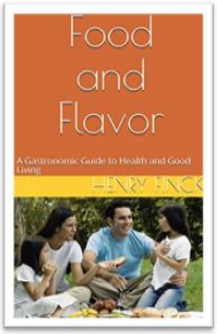
The Gutenberg dataset represents a corpus of over 15,000 book texts, their authors, and titles. The data has been scraped from the Project Gutenberg website using a custom script to parse all bookshelves. we have taken five samples of Gutenberg digital books that are of five different genres. The books are 1. Politics*: Democracy In America by Alexis de Toqueville*. 2. Music: *The Lighter Classics in Music by David Ewen*. 3. Food: *Food and Flavor by Henry Theophilus*, 4. Computer and Tech: The Jargon File by various authors and the last book is Law: The Common Law by Edward Gibbon.

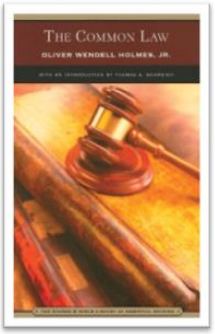
We attempt to use two similar genres *Law and Politics* to make it a little bit challenging for the model to cluster them and this will provide us with a firm ground to evaluate and analysis their performance

Books









Data Preparation:

- First step is reading the books from Gutenberg's library**.**
- Data Preparation:
  - Removing stop-words and garbage characters using regular expression and genism library, also this pattern "[a-zA-Z]{3,}" ensures that words with less than 3 characters like "ye" are removed as they have no special meaning.
  - Converting all words to the lower case to avoid redundancy.
  - Extracting Bigrams and Trigrams using gensim's *Phrases* model. it preserves the meaning of words and sentences.
  - Lemmatization is the next step as it carries out the morphological analysis of the words. this task has been done in the best way using space library.
- Data Partitioning, partition each book into 200 documents, each document is a 150-word record.
- Data labeling as follows:
  1. Politics → a
  2. Law → b
  3. Music → c
  4. Food → d
  5. Computer → e

Each author, Book Title, label and 150-word Records are combined in data frame as follows.

	index	Authors	title	label	a150_Words
138	1	Oliver Wendell Holmes	The Common Law	b	occasioned omission cause act defendant reason...
131	0	Alexis de Toqueville	Democracy In America	a	prejudicial influence fix principle government...
183	4	Various	The Jargon File	e	green one see vanilla cause flavorful usually ...
195	4	Various	The Jargon File	e	foo thing crock people somewhere similar bagbi...
48	2	David Ewen	The Lighter Classics in Music	c	always sensitive physique poor health chopin s...

Feature Engineering and Methodology

Text transformation using BOW, TF-IDF, N-gram, Word Embedding (Word2Vec) and LDA.

- **BOW**: It is a representation of text that describes the occurrence of words within a document, it involves two things:

    1. Vocabulary of known words.
    2. Measurement of the presence of known words.

	abandon	abandoned	abandoning	abandonment	abashed	abate	abatement	abbacy	abbess	abbey	abbreviation	abc	abduction	aberdare
0	0	0	0	0	0	0	0	0	0	0	0	0	0	0
1	0	0	0	0	0	0	0	0	0	0	0	0	0	0
2	0	0	0	0	0	0	0	0	0	0	0	0	0	0
3	0	0	0	0	0	0	0	0	0	0	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0	0	0	0	0	0	0	0	0	0	0	0	0	0
996	0	0	0	0	0	0	0	0	0	0	0	0	0	0
997	0	0	0	0	0	0	0	0	0	0	0	0	0	0
998	0	0	0	0	0	0	0	0	0	0	0	0	0	0
999	0	0	0	0	0	0	0	0	0	0	0	0	0	0

- **TF-IDF:** a technique to quantify words in a set of documents. We compute a score for each word to signify its importance in the document and corpus.

	abandon	abandoned	abandoning	abandonment	abashed	abate	abatement	abbacy	abbess	abbey	abbreviation	abc	abduction	aberdare
0	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
1	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
2	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
3	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
4	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
995	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
996	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
997	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
998	0.112809	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
999	0.000000	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0

- **Word Embedding (Word2Vec)**

It is a vector representation of a particular word, that can capture context of a word in a document, semantic and syntactic similarity, relation with other words.
Word embedding id built using Continuous Skip-Gram Model, the length of the dense vector to represent each token here is 150 matching for each word.

```
[ 2.89420802e-02 -1.01962745e-01  1.21764593e-01 -3.17037851e-01
  6.60284609e-02  1.64465547e-01  6.39201179e-02  8.07203725e-02
  1.08536489e-01  4.08666208e-02 -1.08461075e-01  1.02475286e-01
 -5.61686084e-02  1.11318035e-02 -9.99296680e-02  1.35578305e-01
  8.32538456e-02  1.32464528e-01 -9.66112763e-02  2.18774676e-02
  2.97588762e-02  1.00790970e-01 -3.95088159e-02 -1.50273338e-01
 -2.48950347e-02  8.34373608e-02 -2.77601499e-02 -1.27800226e-01
  1.51634440e-01 -6.73390701e-02  7.42916316e-02  1.31106496e-01
 -2.03051642e-02 -5.85971884e-02 -3.53889130e-02  2.21932933e-01
  1.04234926e-01 -1.29252478e-01  1.74944863e-01 -3.51471007e-02
```

- **LDA:**

It considers each document as a collection of topics in a certain proportion. And each topic as a collection of keywords in a certain proportion. after creating the Dictionary and Corpus needed for Topic Modeling, the model builds the Topic Model using gensim LdaMode.

Number of topics has been used here is 150, although LDA can cluster the documents without even any need to clustering algorithms, but we preferred to use it as a feature engineering and extract 150 topics and let the clustering algorithms complete the job. choosing this number is to make it the same as the number of W2V just to make it easy to evaluate both.

```
(array([[0.0040917 , 0.04198715, 0.00739915, ..., 0.00410041, 2.0261762 ,
         0.0041409 ],
        [0.0040917 , 0.04198715, 0.00739915, ..., 0.00410041, 0.02617621,
         0.0041409 ],
        [0.0040917 , 0.04198715, 0.00739915, ..., 0.00410041, 0.02617621,
         0.0041409 ],
        ...,
        [0.0040917 , 0.04198715, 0.00739915, ..., 0.00410041, 0.02617621,
         0.0041409 ],
        [0.0040917 , 0.04198715, 0.00739915, ..., 0.00410041, 0.02617621,
         0.0041409 ],
        [0.0040917 , 0.04198715, 0.00739915, ..., 0.00410041, 0.02617621,
         0.0041409 ]], dtype=float32), None)
```

Dimensionality Reduction

After text transformation, it is quite hard to deal with this huge number of patterns and here dimensionality reduction comes to just provide us with the most important features without any redundancy. PCA and T-SNE are good choices to use in such situations.

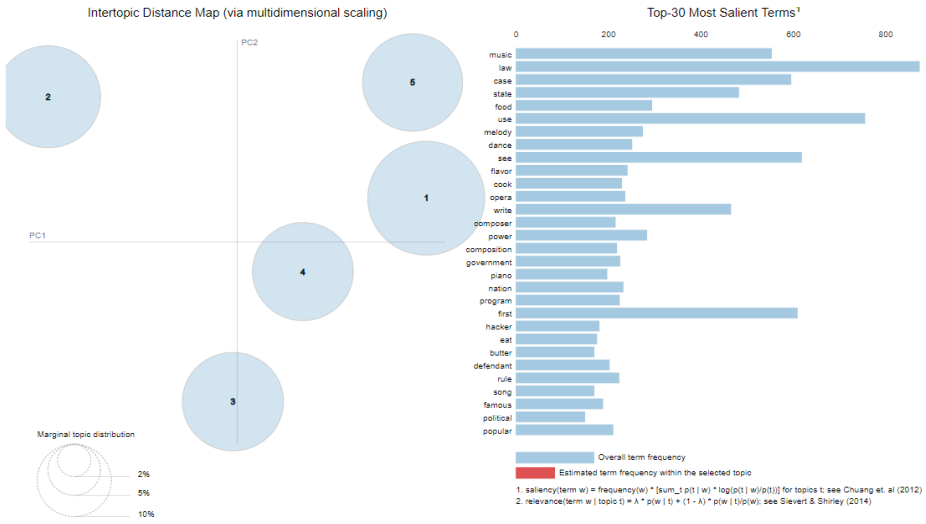
T-SNE differs from PCA by preserving only small pairwise distances or local similarities whereas PCA is concerned with preserving large pairwise distances to maximize variance, so we decide to go with T-SNE as a pre step to the modelling.

Benefits of using T-SNE:

- Reduce dimensions of the data.
- Reduce computation time.
- Good visualizations.

LDA as Topic modelling

LDA tries to locate each word to a specific topic, so if we reduced the number of topics to 5 which is the number of our genres, then LDA will act as a clustering method. as the figure shows below it assign each word to a specific non-overlapping topic.



Clustering

For each technique of the above, these following models are used.

1. K-Means.
2. Expectation Maximization (EM).
3. Hierarchical clustering (Agglomerative).

➢ K-Means

K-Means is one of the simplest and most popular machine learning algorithms out there. It is an unsupervised algorithm as it doesn't use labelled data, in our case it means that no single text belongs to a class or group.

It classifies a dataset into a 5 number of clusters. Euclidean method as a distance metric, k-means++ as our method for initial cluster centres for k-mean clustering in a smart way to speed up convergence and maximum iterations are 300.

- **EM**

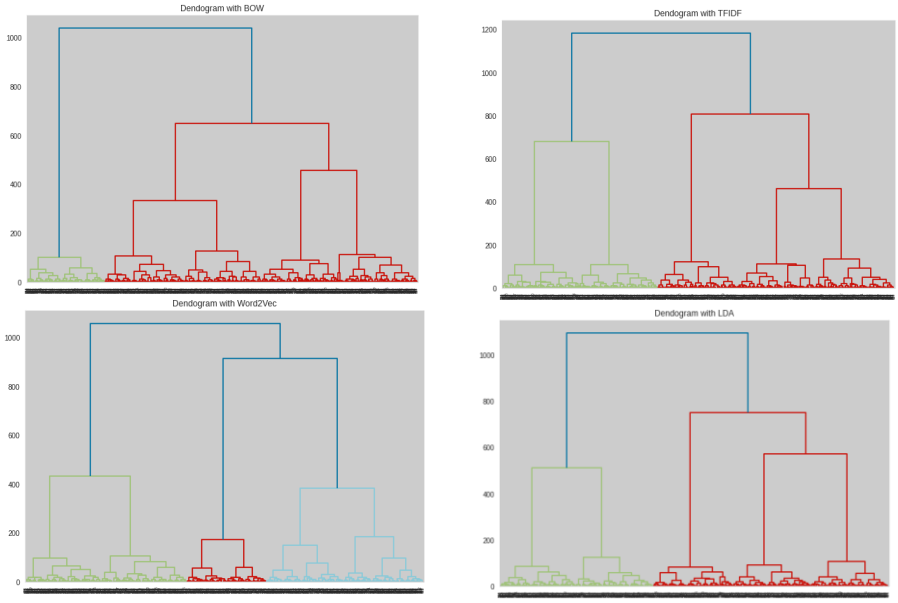
The expectation-maximization algorithm is an approach for performing maximum likelihood estimation in the presence of latent variables. It does this by first estimating the values for the latent variables, then optimizing the model, then repeating these two steps until convergence.

The number of mixture components is equal to the number of clusters (5), covariance type is spherical, so each component has its own single variance and the number of initializations to perform is equal to 10.

- **Hierarchical clustering**

Euclidean metric is used to compute the linkage, The linkage criterion determines which distance to use between sets of observation, so we used ward to minimizes the variance of the clusters being merged.

Here are the dendograms of the Hierarchical clustering with (BOW, TF-IDF, W2V, LDA):



Model Evaluation

We Evaluate our models using different metrics as:

1. Cohen's kappa
2. Silhouette
3. Coherence With LDA

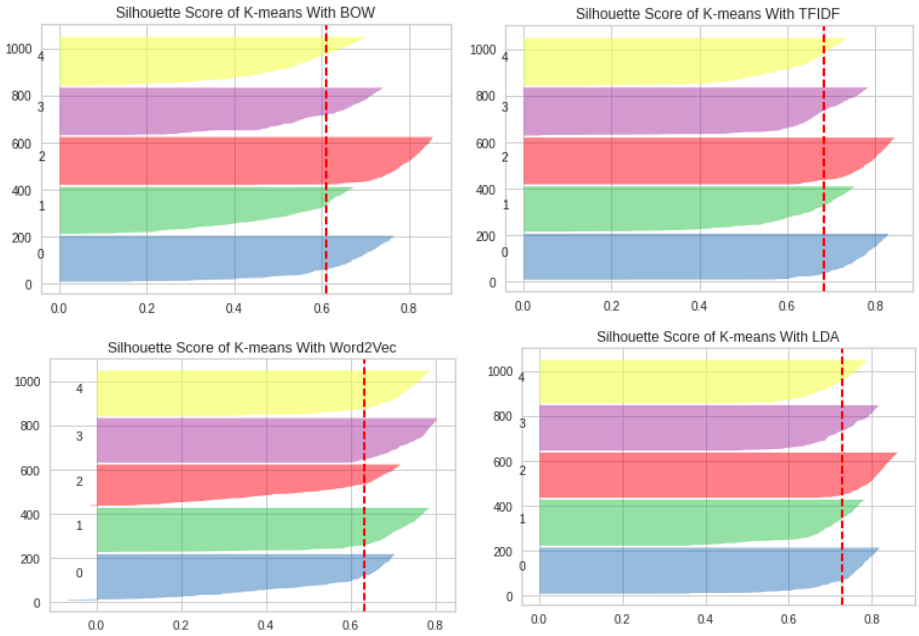
- **Cohen's kappa**
    - It's a statistic that measures inter-annotator agreement
    - We measure it for our 3 models using cohen_kappa_score from sklearn

Kappa	K-means	EM	Hierarchical clustering	LDA modelling
BOW	0.4962	0.2525	-0.0037	0.0467
TF-IDF	0.2512	0.0000	0.0100	
Word2Vec	0.0275	0.0463	0.3113	
LDA	0.4938	0.2500	-0.0025	

- **Silhouette**
    - It's calculated using the mean intra-cluster distance (a) and the mean nearest-cluster distance (b) for each sample
    - We measure it for our 3 models using silhouette_score from sklearn

Silhouette	K-means	EM	Hierarchical clustering
BOW	0.6101	0.6136	0.6075
TF-IDF	0.6842	0.6797	0.6842
Word2Vec	0.6326	0.6274	0.6247
LDA	0.7302	0.7276	0.7302

    - These figures show the Silhouette for K-means with different transformations.



- **Consistency**

  Here, we calculate the *in-between* distance for the 5 clusters and *within* distance for all data points within clusters.

  1. The within distance calculated between each data point and its cluster centroid
     - The lower the distance, the better the model
  2. The in-between distance is calculated as the distance between all centroids of the 5 clusters
     - The higher the distance, the better the model

BOW K-means Consistency

```
Distance between clusters:

[[ 0.       71.38036  51.956654 29.49012  32.010406]
 [71.38036   0.       63.457104 67.53231  40.874454]
 [51.956654 63.457104  0.       23.696224 37.74758 ]
 [29.49012  67.53231  23.696224  0.       29.577686]
 [32.010406 40.874454 37.74758  29.577686  0.      ]]

Distance within clusters:

[7.297173299726937, 6.986720539252651, 8.967613691321235, 7.615792774690165, 8.315876837719271]
```

TFIDF K-means Consistency

```
Distance between clusters:

[[ 0.       65.2866   48.19853  39.9811   61.42686 ]
 [65.2866    0.       85.22971  32.543587 45.394176]
 [48.19853  85.22971   0.       76.20623  51.42725 ]
 [39.9811   32.543587 76.20623   0.       57.3762  ]
 [61.42686  45.394176 51.42725  57.3762    0.      ]]

Distance within clusters:

[7.974990932962928, 9.002834236350322, 7.77589499018522, 9.146748226515362, 9.280678174732552]
```

Embedding K-means Consistency

```
Distance between clusters:

[[ 0.       79.98708  64.533676 53.3711   49.534184]
 [79.98708   0.       62.550274 30.259819 41.599495]
 [64.533676 62.550274  0.       61.94725  27.78911 ]
 [53.3711   30.259819 61.94725   0.       34.262016]
 [49.534184 41.599495 27.78911  34.262016  0.      ]]

Distance within clusters:

[9.033199532656845, 7.232376538980244, 8.515434099492515, 7.204983346376152, 8.774780645824823]
```

LDA K-means Consistency

```
Distance between clusters:

[[ 0.       70.12624  49.880177 50.24168  41.29125 ]
 [70.12624   0.       78.80313  38.819244 36.302963]
 [49.880177 78.80313   0.       40.35502  71.94617 ]
 [50.24168  38.819244 40.35502   0.       43.87971 ]
 [41.29125  36.302963 71.94617  43.87971   0.      ]]

Distance within clusters:

[6.452366239723695, 7.567407254673963, 6.425996513832602, 6.772310512603615, 6.481546439603520 5]
```

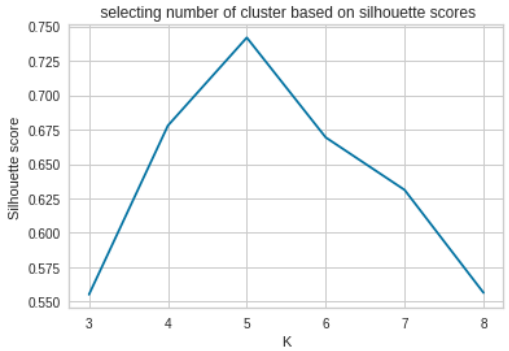
- **Coherence With LDA**
  - It's used to measure how well the topics are extracted
  - Coherence score With LDA using u-mass method: **-8.0875**
    - **Lower is better**
  - Coherence score With LDA using c_v method : **0.4621**
    - **Higher is better**

Error Analysis

Silhouette coefficient ranges from -1 to +1, high coefficient value indicates that the object is well matched to its own cluster and poorly matched to neighboring clusters, while a coefficient close to 0 indicates that the sample is on or very close to the decision boundary, finally a value close to -1 indicates that those samples might have been assigned to the wrong cluster.

```
For n_clusters = 3 The average silhouette_score is: 0.55520874
For n_clusters = 4 The average silhouette_score is: 0.67790955
For n_clusters = 5 The average silhouette_score is: 0.7418159
For n_clusters = 6 The average silhouette_score is: 0.6693198
For n_clusters = 7 The average silhouette_score is: 0.631178
For n_clusters = 8 The average silhouette_score is: 0.55648196
```

The silhouette scores confirm that the K= 5 is the optimal value for number of clusters that matches well with the actual number of topics, also underlines the fact that K = 4 and K = 6 are quite good as a second choice.

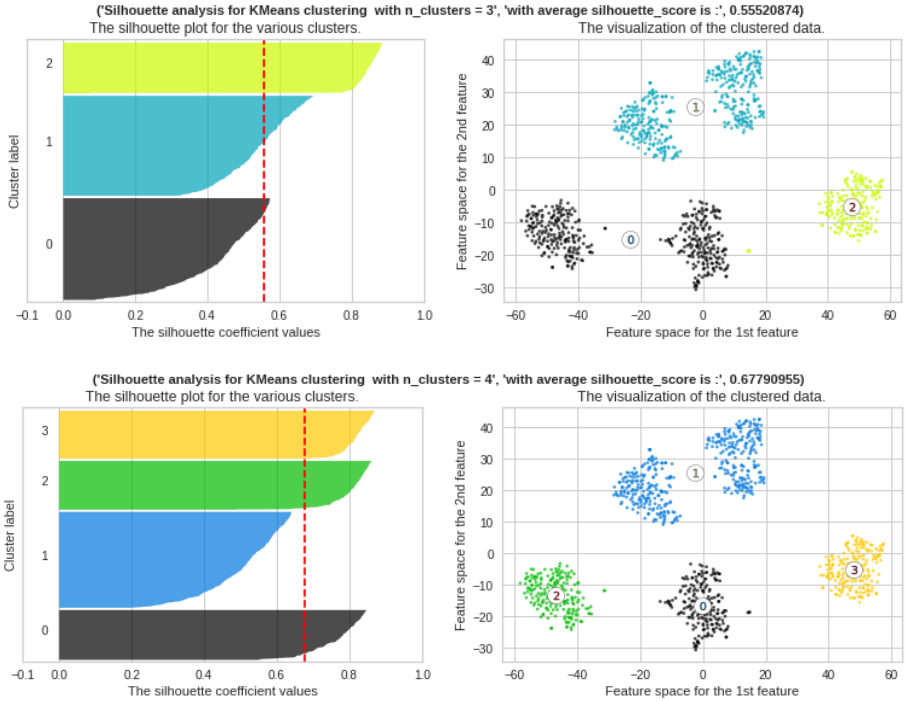


By plotting the silhouette diagram which contains one knife shape per cluster. The shape's height indicates the number of instances the cluster contains, and its width represents the sorted silhouette coefficients of the instances in the cluster, the dashed line indicates the mean silhouette coefficient.

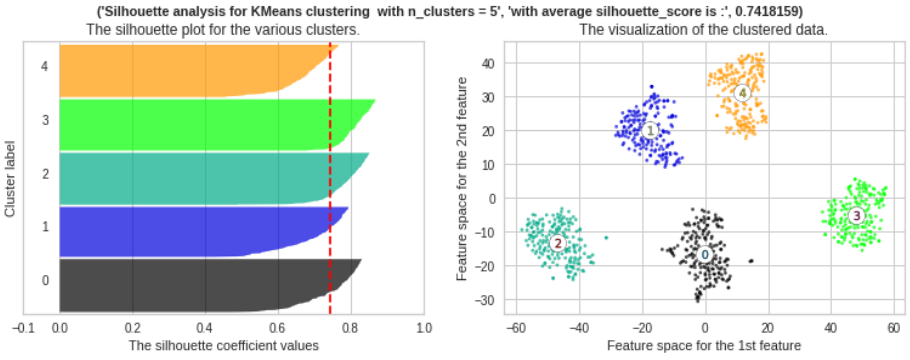
When most of the instances in a cluster have a lower coefficient than the average silhouette score, then the cluster is rather bad since this means its instances are much too close to the other clusters rather than its assigned cluster.

The following figures indicate the silhouette diagram for various values of K, and by understanding their results we can decide which k value can reduce the machine's error as possible and increase the model performance.

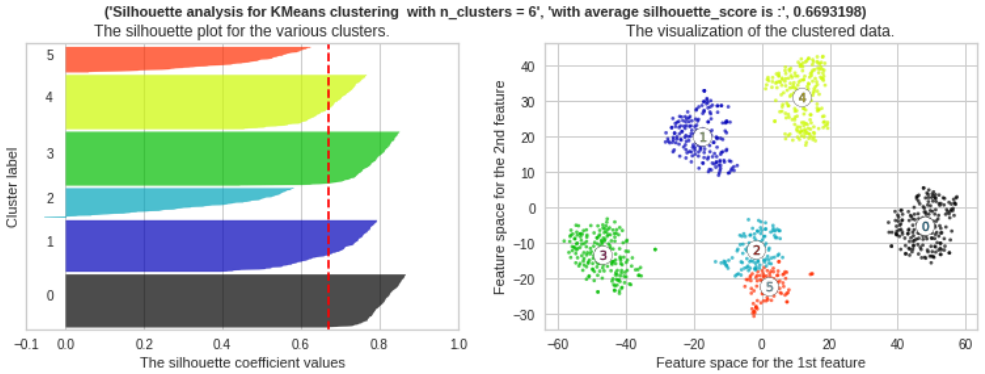
- At k = 3 and k = 4 we get a bad cluster while they have high average silhouette scores because clusters 0 and 0 for k = 3 and cluster 1 for k = 4 have a high number of instances and most of them stop short of the dashed line, ending to the left of it which means most of this instance in that cluster have a lower coefficient than average silhouette score so we can't use 3 or 4 as the number of clusters



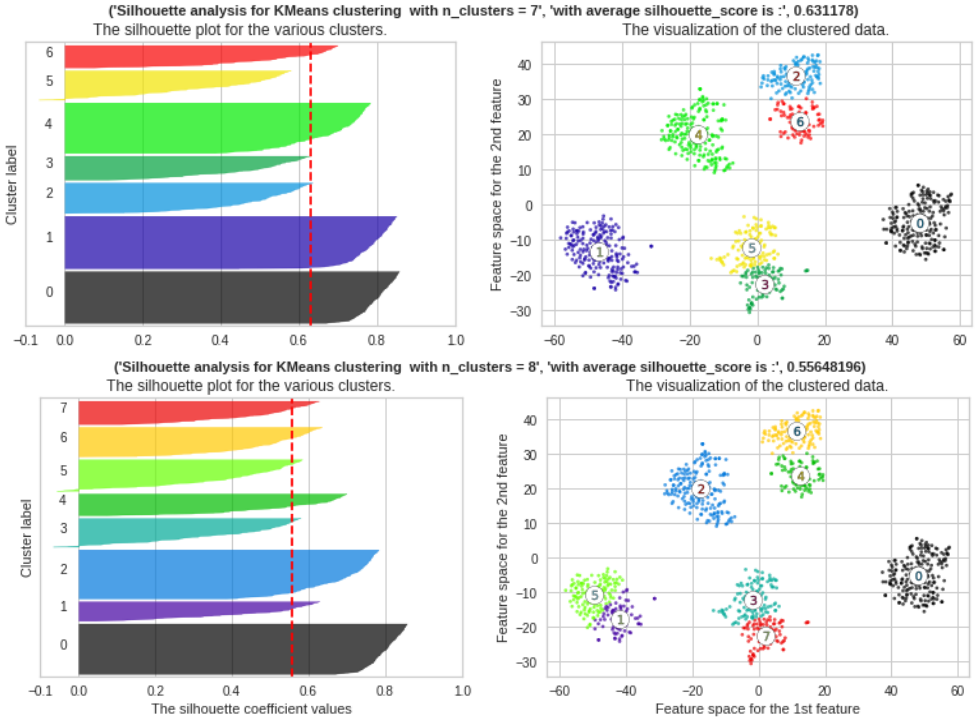
- At k = 5, we notice that all clusters have similar sizes and clusters looks pretty good as most instances extend beyond the dashed line, to the right, and closer to 1 as clusters have an average silhouette of about 0.7 which indicate that the clusters had separated well and most of the instances in the same are very similar to each other.



- when K = 6 we get a similar result as k = 5, but we notice that the cluster 0 in K = 5 is split into two clusters (cluster 2 and cluster 5) and each of them has a small cluster size and all their instances have lower silhouette coefficient than the average which indicates that it belongs to other clusters rather than its cluster which decrease the overall average silhouette for the clusters which indicate that the K of instances = 6 is not a good choice for the problem case.



- At k = 7 and k = 8 we found that only clusters 0,1 and 4 for K = 7 and clusters 0 and 2 for K = 8 have a stable silhouette score and size while all instances in other clusters have a lower silhouette coefficient than the average which indicates that it belongs to other clusters rather than its cluster and a so, we can't use 7 or 8 as the number of clusters.



As the optimal number of clusters that the machine recognized is equal to the actual number of topics, we have ensured that the clustering model passed the tricky step successfully.

Once we have a promising model (which in our case K Means with LDA as a feature engineering). Now we will try to figure out what makes the machine confused.

If we showed the top 20 frequent words in clusters and label and tried to find the matches.

```
labels_mf , labels_mfw  = most_frequent_words (20,
                                               analysis_df,
                                               "index",
                                               "a150_Words")

clusters_mf,clusters_mfw = most_frequent_words( 20,
                                               analysis_df,
                                               "cluster_output",
                                               "a150_Words")

lab_cls_match = matches_dict(labels_mf , clusters_mf)
lab_cls_match
```

```
{0: [9, 7, 8, 8, 8],
 1: [6, 7, 7, 6, 6],
 2: [7, 7, 6, 8, 8],
 3: [7, 7, 7, 7, 7],
 4: [9, 9, 10, 8, 9]}
```

We found that documents labeled as 0 match with 9 words out of 20 with the first cluster, 7 words with the second cluster, and 8 words for each other clusters, so each of the clusters has a similar number of words that matches with the label.

The next line of code indicates how much these words occurred in the cluster, as we see the 9 words that match between label 0 and the first cluster occurred 1237 times while the 8 words match the label 0 with the third cluster occurred 1592 so we can't judge such words as its unique values occurred more in a cluster, but its frequent values are more in other clusters. so that is why we need to find these words.

```
lab_cls_weights = weights_dict(labels_mf ,
                               clusters_mf ,
                               clusters_mfw)
lab_cls_weights
```

```
{0: [1237, 1325, 1066, 1592, 1203],
 1: [904, 1325, 962, 1259, 969],
 2: [1017, 1325, 857, 1592, 1203],
 3: [1017, 1325, 962, 1429, 1087],
 4: [1237, 1618, 1266, 1592, 1318]}
```

we need to dive deep to figure out these words by showing the most frequent 20 words in each label (Ground Truth) and the most frequent 20 words of each cluster.

the top frequent 20 words for the labels are:

most frequent words in label: 0

[('make', 358), ('food', 300), ('good', 248), ('cook', 246), ('flavor', 243), ('eat', 196), ('butter', 185), ('meat', 168), ('use', 159), ('much', 159), ('many', 151), ('may', 137), ('give', 133), ('would', 129), ('time', 126), ('know', 121), ('way', 117), ('french', 114), ('take', 98), ('great', 96)]

most frequent words in label: 1

[('music', 591), ('first', 419), ('write', 329), ('melody', 301), ('dance', 282), ('work', 242), ('opera', 233), ('composer', 221), ('composition', 217), ('piano', 211), ('come', 209), ('make', 196), ('popular', 189), ('year', 188), ('become', 187), ('song', 185), ('famous', 167), ('also', 156), ('bear', 156), ('die', 144)]

most frequent words in label: 2

[('state', 374), ('law', 334), ('may', 307), ('power', 233), ('government', 221), ('nation', 219), ('people', 205), ('great', 194), ('man', 171), ('political', 154), ('country', 142), ('would', 138), ('time', 128), ('interest', 122), ('right', 121), ('influence', 117), ('public', 117), ('see', 115), ('authority', 110), ('citizen', 109)]

most frequent words in label: 3

[('use', 490), ('see', 402), ('also', 282), ('program', 210), ('hacker', 205), ('term', 165), ('may', 158), ('system', 156), ('often', 140), ('file', 139), ('computer', 131), ('bit', 126), ('time', 126), ('name', 123), ('common', 122), ('make', 114), ('example', 110), ('character', 109), ('say', 106), ('compare', 101)]

most frequent words in label: 4

[('law', 541), ('case', 409), ('may', 334), ('would', 287), ('man', 265), ('act', 224), ('say', 217), ('defendant', 217), ('fact', 202), ('make', 201), ('must', 174), ('take', 166), ('rule', 165), ('possession', 165), ('action', 163), ('liability', 148), ('know', 144), ('intent', 142), ('find', 141), ('good', 137)]

the top frequent 20 words for the label clusters are:

most frequent words in cluster: 0

[('law', 179), ('may', 177), ('make', 170), ('use', 138), ('man', 123), ('first', 117), ('state', 113), ('case', 111), ('also', 109), ('time', 107), ('give', 106), ('would', 101), ('see', 101), ('music', 96), ('take', 91), ('many', 84), ('come', 82), ('good', 81), ('say', 81), ('work', 81)]

most frequent words in cluster: 1

[('make', 221), ('may', 213), ('use', 205), ('first', 198), ('law', 171), ('would', 159), ('see', 158), ('also', 147), ('music', 146), ('man', 137), ('time', 136), ('good', 128), ('write', 122), ('case', 122), ('come', 120), ('give', 118), ('say', 106), ('state', 105), ('must', 105), ('work', 104)]

most frequent words in cluster: 2

[('may', 176), ('make', 156), ('law', 147), ('use', 146), ('would', 125), ('see', 107), ('first', 105), ('man', 104), ('also', 101), ('case', 99), ('music', 98), ('time', 96), ('good', 94), ('write', 93), ('say', 90), ('know', 85), ('work', 72), ('give', 72), ('find', 72), ('become', 71)]

most frequent words in cluster: 3

[('law', 250), ('make', 228), ('use', 219), ('may', 213), ('would', 177), ('case', 172), ('time', 170), ('see', 163), ('first', 162), ('music', 147), ('also', 137), ('give', 132), ('good', 123), ('state', 122), ('say', 122), ('man', 120), ('write', 116), ('part', 115), ('become', 112), ('great', 110)]

most frequent words in cluster: 4

[('may', 205), ('make', 192), ('law', 178), ('use', 136), ('see', 134), ('case', 124), ('first', 118), ('also', 116), ('state', 115), ('music', 110), ('time', 106), ('would', 94), ('good', 92), ('say', 92), ('write', 90), ('great', 87), ('come', 86), ('must', 85), ('man', 84), ('many', 84)]

May				***Make***			
labels		***clusters***		***labels***		***clusters***	
0	***137***	***0***	***170***	***0***	***358***	***0***	***170***
1	***-***	***1***	***213***	***1***	***196***	***1***	***221***
2	***307***	***2***	***176***	***2***	***-***	***2***	***156***
3	***185***	***3***	***213***	***3***	***114***	***3***	***228***
4	***334***	***4***	***205***	***4***	***201***	***4***	***192***

So, words like “may” and “make” cause a big confusion as they are considered as top frequent words in all labels while it is a neutral word and have no bias towards a specific genre, including such words in stop words may help in reducing the level of confusion which threw the machine off.

The other thing that may cause errors is that if a marker word for a specific label (like law word which is a marker word for The Common Law book) is identified as a one of the top frequent words in other label (law word is a frequent word in Democracy in America book)

While Politics and Law are interconnected, they’re not the same thing. and as it is hard for humans to perfectly identify the real genre of interconnected topics, the machine has done the job in a very satisfying way.

Using LDA as feature engineering is quite different from using it as topic modeling or clustering techniques, LDA assigns one topic to each word so we can say, it is clustered based on word's weight or word's topic, but as feature engineering, we represent each word by a number from 1 to 150 which is the number of our topics and cluster based on the document, not the words separately, so documents are clustered together based on the weights of their words together and as the kappa score is higher here then this approach is better to be used, so we decided to use LDA as feature engineering

Using LDA as feature engineering gives the best silhouette values through the three models and this is close to the human decision, the latent meaning laying into the word sometimes is more important than how many times you said the word, as we said before the word has a value and, in this case, it takes its value from its implicit meaning.

References

https://machinelearningmastery.com/expectation-maximization-em-algorithm/

https://www.dcs.bbk.ac.uk/~Dell/teaching/cc/book/ditp/ditp_ch7.pdf