Our business case is to predict most valuable travel book based on user preference We choose 5 books from Gutenberg library and they are:

1 – "A London Life" which was written by the author "Henry James", we label it in our dataframe as "c", you can find the book in the following link:

https://www.gutenberg.org/files/25500/25500-8.txt

2 – "Malay Magic" which was written by the author "Walter William Skeat", we label it in our dataframe as "d", you can find the book in the following link: https:

//www.gutenberg.org/files/47873/47873-8.txt

3 – "Chicago by day and night" which was written by anonymous author, we label it in our dataframe as "b", you can find the book in the following link:

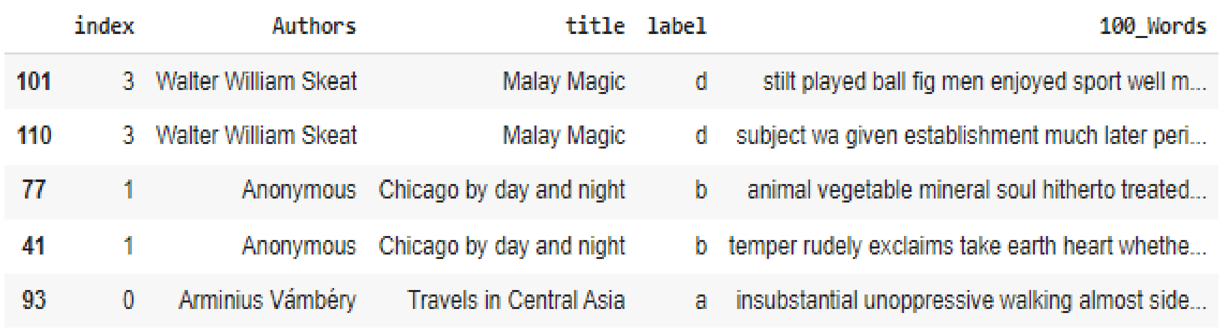
https://www.gutenberg.org/files/70675/70675-0.txt

4 – "The Private Life of the Romans" which was written by the author "Harold Whetstone Johnston", we label it in our dataframe as "e", you can find the book in the following link: https://www.gutenberg.org/files/40549/40549-0.txt

5 – "Travels in Central Asia" which was written by the author "Arminius Vámbéry", we label it in our dataframe as "a", you can find the book in the following link: https://www.gutenberg.org/files/41751/41751-8.txt

In data preprocessing we remove stop words that don't really signify any importance and don't distinguish the books, then we did lemmatization to convert the words to its meaningful base form, we also did lower casing to convert the word to its lower case for simplicity.

After that we create samples of 200 documents (200 partition), each partition has 150 words The output of the preprocessing process is:

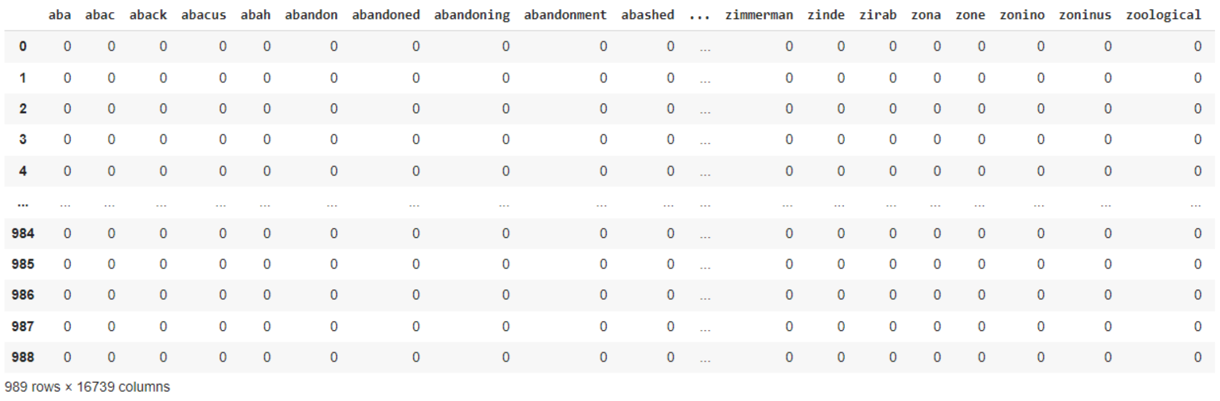


	index	Authors	title	label	100_Words
101	3	Walter William Skeat	Malay Magic	d	stilt played ball fig men enjoyed sport well m...
110	3	Walter William Skeat	Malay Magic	d	subject wa given establishment much later peri...
77	1	Anonymous	Chicago by day and night	b	animal vegetable mineral soul hitherto treated...
41	1	Anonymous	Chicago by day and night	b	temper rudely exclaims take earth heart whethe...
93	0	Arminius Vámbéry	Travels in Central Asia	a	insubstantial unoppressive walking almost side...

Feature engineering

The process of transforming raw data into features that can be used to train machine learning models. One common technique for feature engineering is to use bag-of-words (BOW) and term frequency-inverse document frequency (TF-IDF) representations we use in our model:

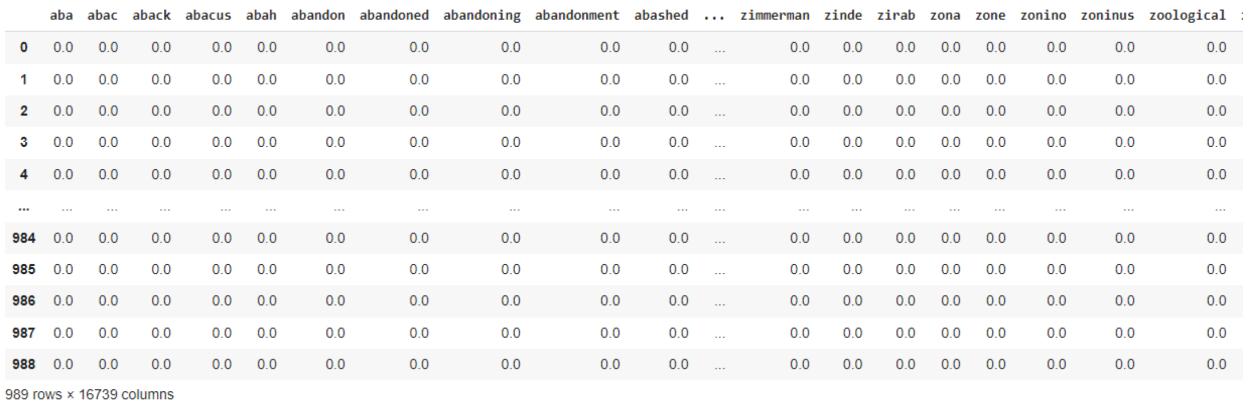
1- BOW: creates a matrix of word counts for each document in a corpus. The CountVectorizer() function from the Scikit-learn library can be used to perform this transformation. The output of the code:



	aba	abac	aback	abacus	abah	abandon	abandoned	abandoning	abandonment	abashed	...	zimmerman	zinde	zirab	zona	zone	zonino	zoninus	zoological
0	0	0	0	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	0	0	0	0
2	0	0	0	0	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	0	0	0	0
4	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
984	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
985	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
986	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
987	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0
988	0	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0

989 rows × 16739 columns

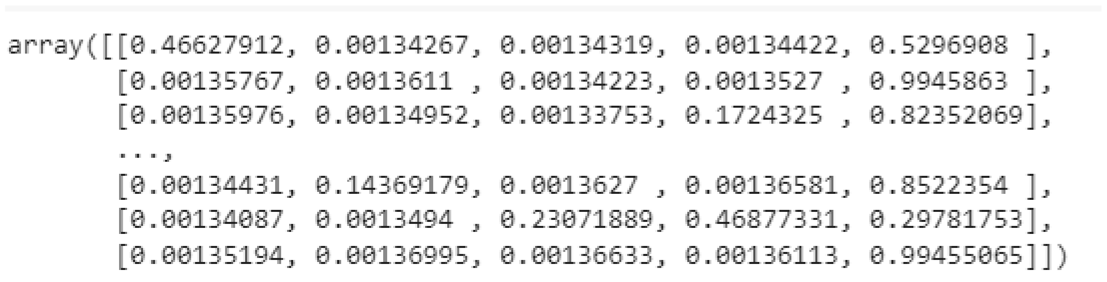
2- -TF-IDF: is a technique that weights the importance of each word in a document based on its frequency in the document and its frequency in the entire corpus. We used TF-IDF after BOW for feature engineering because it takes into account the importance of each word in the document and in the corpus, which can lead to better performance in machine learning models. The output of the code



	aba	abac	aback	abacus	abah	abandon	abandoned	abandoning	abandonment	abashed	...	zimmerman	zinde	zirab	zona	zone	zonino	zoninus	zoological
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	0.0	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	0.0	0.0	0.0	...	0.0	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	0.0	0.0	0.0	...	0.0	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	0.0	0.0	0.0	...	0.0	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	0.0	0.0	0.0	...	0.0	0.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...	...
984	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	0.0	0.0	0.0	0.0	0.0
985	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	0.0	0.0	0.0	0.0	0.0
986	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	0.0	0.0	0.0	0.0	0.0
987	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	0.0	0.0	0.0	0.0	0.0
988	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	0.0	0.0	0.0	0.0	0.0

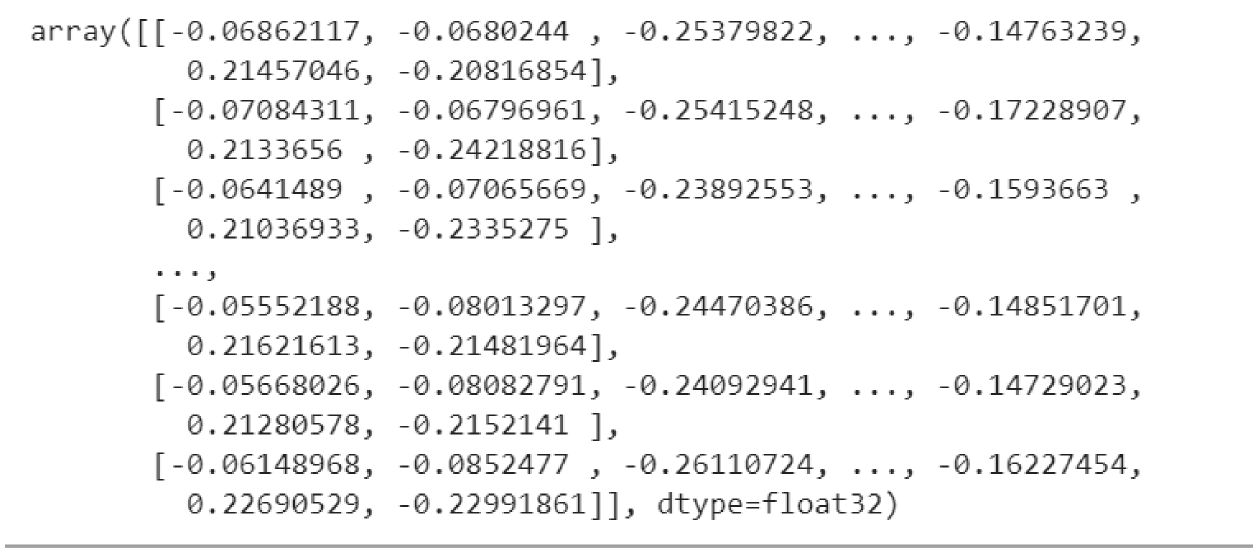
989 rows × 16739 columns

3- LDA: which involves identifying the main themes or topics that occur in a collection of documents. The output of LDA can be used to summarize the content of the documents and to gain insights into the underlying themes and patterns in the data. And this is the output from code:



```
array([[0.46627912, 0.00134267, 0.00134319, 0.00134422, 0.5296908 ],
       [0.00135767, 0.0013611 , 0.00134223, 0.0013527 , 0.9945863 ],
       [0.00135976, 0.00134952, 0.00133753, 0.1724325 , 0.82352069],
       ...,
       [0.00134431, 0.14369179, 0.0013627 , 0.00136581, 0.8522354 ],
       [0.00134087, 0.0013494 , 0.23071889, 0.46877331, 0.29781753],
       [0.00135194, 0.00136995, 0.00136633, 0.00136113, 0.99455065]])
```

4- word embeddings : are created using the Word2Vec model from the gensim library. Word embeddings are a type of distributed representation of words in a high-dimensional space, where each word is represented by a dense vector of real numbers. The Word2Vec model is trained on the text data using the splitted_sent input, which is assumed to be a list of sentences. The model learns to associate words that appear in similar contexts and generates word vectors that capture the semantic and syntactic relationships between the words. And this is the output from code:



```
array([[-0.06862117, -0.0680244 , -0.25379822, ..., -0.14763239,
         0.21457046, -0.20816854],
       [-0.07084311, -0.06796961, -0.25415248, ..., -0.17228907,
         0.2133656 , -0.24218816],
       [-0.0641489 , -0.07065669, -0.23892553, ..., -0.1593663 ,
         0.21036933, -0.2335275 ],
       ...,
       [-0.05552188, -0.08013297, -0.24470386, ..., -0.14851701,
         0.21621613, -0.21481964],
       [-0.05668026, -0.08082791, -0.24092941, ..., -0.14729023,
         0.21280578, -0.2152141 ],
       [-0.06148968, -0.0852477 , -0.26110724, ..., -0.16227454,
         0.22690529, -0.22991861]], dtype=float32)
```

Modeling:

1- KMeans algorithm

we applied KMeans algorithm to cluster data on four different representations: bag-of-words (BOW), TF-IDF, LDA, and word embeddings. The KMeans algorithm is an unsupervised learning algorithm that groups data points into clusters based on their similarity. The number of clusters is specified by the n_clusters parameter.

2- EM algorithm

EM stands for Expectation-Maximization, which is a clustering algorithm used to estimate the parameters of a probabilistic model. In this case, the GaussianMixture function from the scikit-learn library is used to perform EM clustering. we applied EM algorithm to cluster data on four different representations: bag-of-words (BOW), TF-IDF, LDA, and word embeddings.

3- Hierarchical clustering algorithm

Hierarchical clustering is a clustering technique that seeks to build a hierarchy of nested clusters. In hierarchical clustering, the data points are successively grouped into clusters based on their similarity or dissimilarity. The result of hierarchical clustering is a dendrogram, which is a tree-like diagram that shows the relationships between the clusters at different levels of the hierarchy. we applied Hierarchical algorithm to cluster data on four different representations: bag-of-words (BOW), TF-IDF, LDA, and word embeddings.

Evaluation:

1- Kappa

Kappa, is a statistical measure of inter-rater agreement that takes into account the possibility of agreement occurring by chance. Kappa ranges from -1 to 1, with a value of 1 indicating perfect agreement, 0 indicating agreement no better than chance, and -1 indicating perfect disagreement. To calculate Cohen's Kappa, you need to have two sets of categorical ratings or labels from two different raters or observers. The Kappa coefficient measures the proportion of agreement between the two raters beyond what would be expected by chance.

2- Coherence

Coherence is a measure of how interpretable and meaningful the topics discovered by a topic modeling algorithm are. It is used to evaluate the quality of the topics generated by the algorithm and to select the optimal number of topics to use in the model . Coherence is a measure of the interpretability and semantic consistency of the topics generated by a topic modeling algorithm. It measures the degree to which the words in each topic are related to each other and how distinct the topics are from one another. In other words, coherence measures how well the topics make sense and how easily they can be understood by a human.

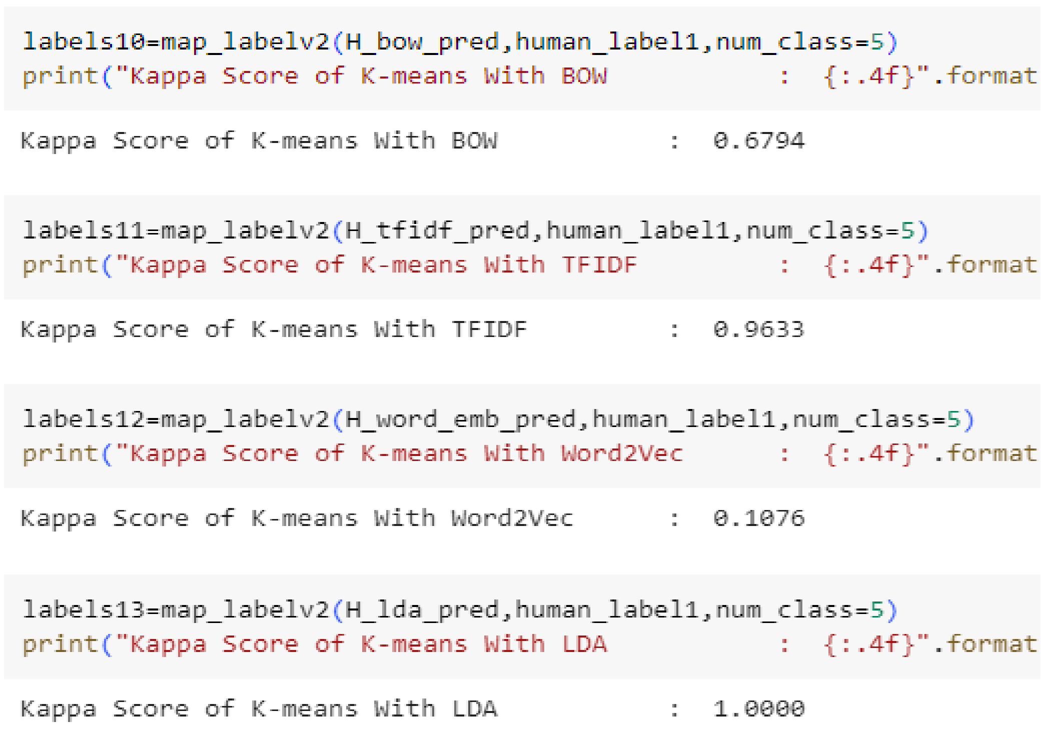
3- Silhouette

Silhouette is a metric used to evaluate the quality of clustering results. It measures the similarity of data points within a cluster compared to points in other clusters. It takes into account both the cohesion (how close the data points are to their own cluster) and separation (how far the data points are from other clusters) of the clustering result.The Silhouette score ranges from -1 to 1, where a score of 1 indicates that the data point is well-matched to its own cluster and poorly matched to neighboring clusters, a score of 0 indicates that the data point is on the boundary between two clusters, and a score of -1 indicates that the data point is poorly matched to its own cluster and well-matched to neighboring cluster.The Silhouette method is useful for determining the optimal number of clusters to use in a clustering algorithm, as well as for comparing the quality of different clustering algorithms. A higher Silhouette score indicates better cluster separation and cohesion, indicating a better clustering result.

Champion model

Champion model is the hierarchical clustering model with Word embedding has a higher silhouette score compared to other models, it is likely that it produces better quality clusters. This means that the model is able to group similar data points together and separate dissimilar data points into different clusters.

Testing Result



```
labels10=map_labelv2(H_bow_pred,human_label1,num_class=5)
print("Kappa Score of K-means With BOW          :  {:.4f}".format
```

```
Kappa Score of K-means With BOW          :  0.6794
```

```
labels11=map_labelv2(H_tfidf_pred,human_label1,num_class=5)
print("Kappa Score of K-means With TFIDF        :  {:.4f}".format
```

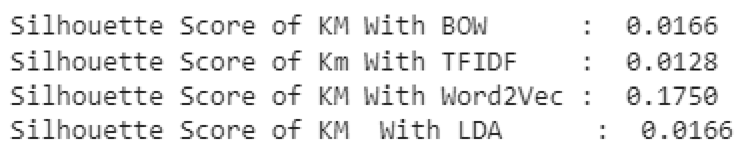
```
Kappa Score of K-means With TFIDF        :  0.9633
```

```
labels12=map_labelv2(H_word_emb_pred,human_label1,num_class=5)
print("Kappa Score of K-means With Word2Vec     :  {:.4f}".format
```

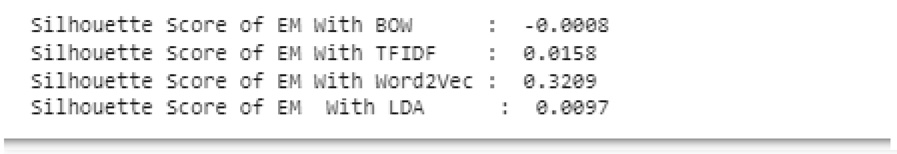
```
Kappa Score of K-means With Word2Vec     :  0.1076
```

```
labels13=map_labelv2(H_lda_pred,human_label1,num_class=5)
print("Kappa Score of K-means With LDA          :  {:.4f}".format
```

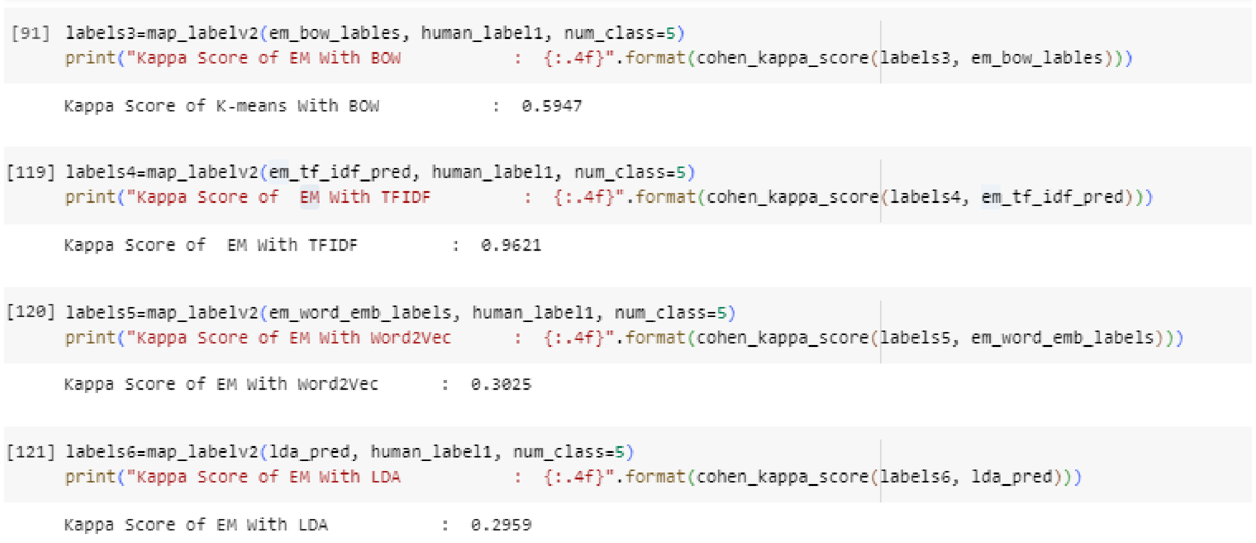
```
Kappa Score of K-means With LDA          :  1.0000
```



```
Silhouette Score of KM With BOW      :  0.0166
Silhouette Score of Km With TFIDF    :  0.0128
Silhouette Score of KM With Word2Vec :  0.1750
Silhouette Score of KM  With LDA      :  0.0166
```

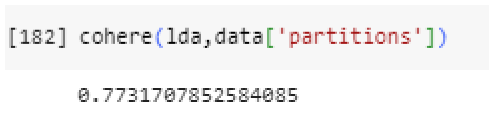


```
Silhouette Score of EM With BOW      :  -0.0008
Silhouette Score of EM With TFIDF    :  0.0158
Silhouette Score of EM With Word2Vec :  0.3209
Silhouette Score of EM  With LDA      :  0.0097
```

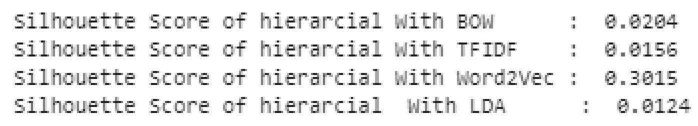


```
[91] labels3=map_labelv2(em_bow_lables, human_label1, num_class=5)
     print("Kappa Score of EM With BOW           :  {:.4f}".format(cohen_kappa_score(labels3, em_bow_lables)))

     Kappa Score of K-means With BOW           :  0.5947

[119] labels4=map_labelv2(em_tf_idf_pred, human_label1, num_class=5)
      print("Kappa Score of  EM With TFIDF         :  {:.4f}".format(cohen_kappa_score(labels4, em_tf_idf_pred)))

     Kappa Score of  EM With TFIDF         :  0.9621

[120] labels5=map_labelv2(em_word_emb_labels, human_label1, num_class=5)
      print("Kappa Score of EM With Word2Vec      :  {:.4f}".format(cohen_kappa_score(labels5, em_word_emb_labels)))

     Kappa Score of EM With Word2Vec      :  0.3025

[121] labels6=map_labelv2(lda_pred, human_label1, num_class=5)
      print("Kappa Score of EM With LDA           :  {:.4f}".format(cohen_kappa_score(labels6, lda_pred)))

     Kappa Score of EM With LDA           :  0.2959
```

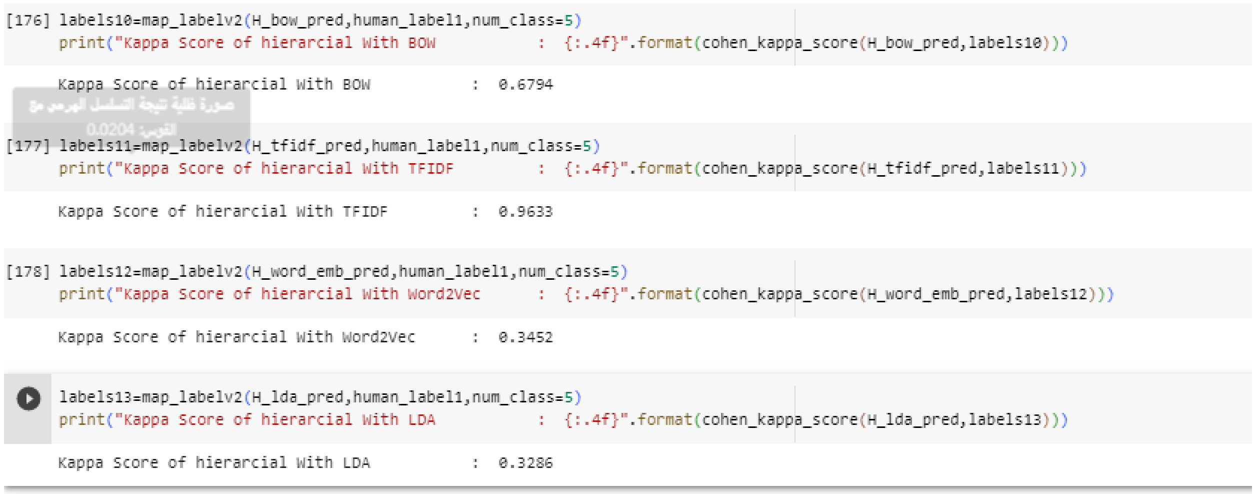
Coherence With LDA



```
[182] cohere(lda,data['partitions'])

      0.7731707852584085
```



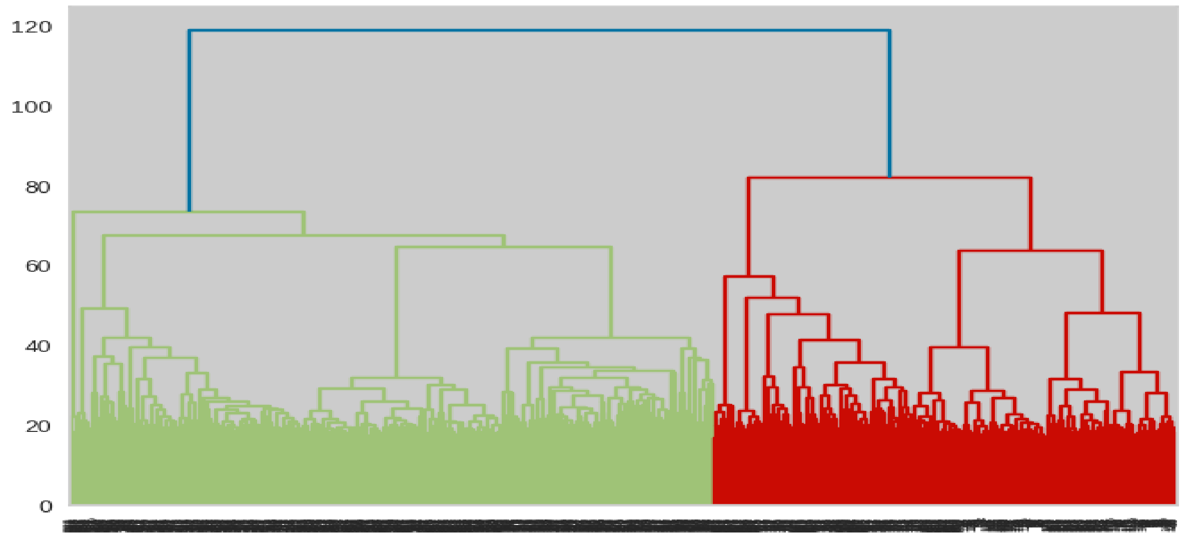
```
Silhouette Score of hierarcial With BOW      :  0.0204
Silhouette Score of hierarcial With TFIDF    :  0.0156
Silhouette Score of hierarcial With Word2Vec :  0.3015
Silhouette Score of hierarcial  With LDA      :  0.0124
```



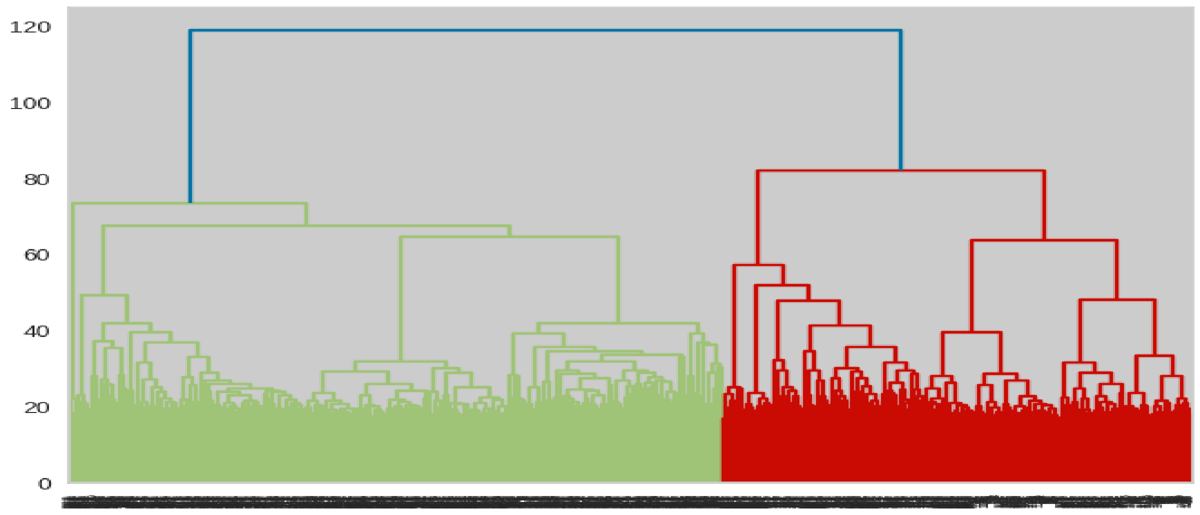
```
[176] labels10=map_labelv2(H_bow_pred,human_label1,num_class=5)
      print("Kappa Score of hierarcial With BOW           :  {:.4f}".format(cohen_kappa_score(H_bow_pred,labels10)))

      Kappa Score of hierarcial With BOW           :  0.6794
صورة ظلية نتيجة التسلسل الهرمي مع
التوسيل: 0.0204

[177] labels11=map_labelv2(H_tfidf_pred,human_label1,num_class=5)
      print("Kappa Score of hierarcial With TFIDF         :  {:.4f}".format(cohen_kappa_score(H_tfidf_pred,labels11)))

      Kappa Score of hierarcial With TFIDF         :  0.9633

[178] labels12=map_labelv2(H_word_emb_pred,human_label1,num_class=5)
      print("Kappa Score of hierarcial With Word2Vec      :  {:.4f}".format(cohen_kappa_score(H_word_emb_pred,labels12)))

      Kappa Score of hierarcial With Word2Vec      :  0.3452

      labels13=map_labelv2(H_lda_pred,human_label1,num_class=5)
      print("Kappa Score of hierarcial With LDA           :  {:.4f}".format(cohen_kappa_score(H_lda_pred,labels13)))

      Kappa Score of hierarcial With LDA           :  0.3286
```

hierarchical clustering on a dataset x using the Ward method and Euclidean distance as the similarity metric. The goal is to group similar data points together into clusters based on their feature values

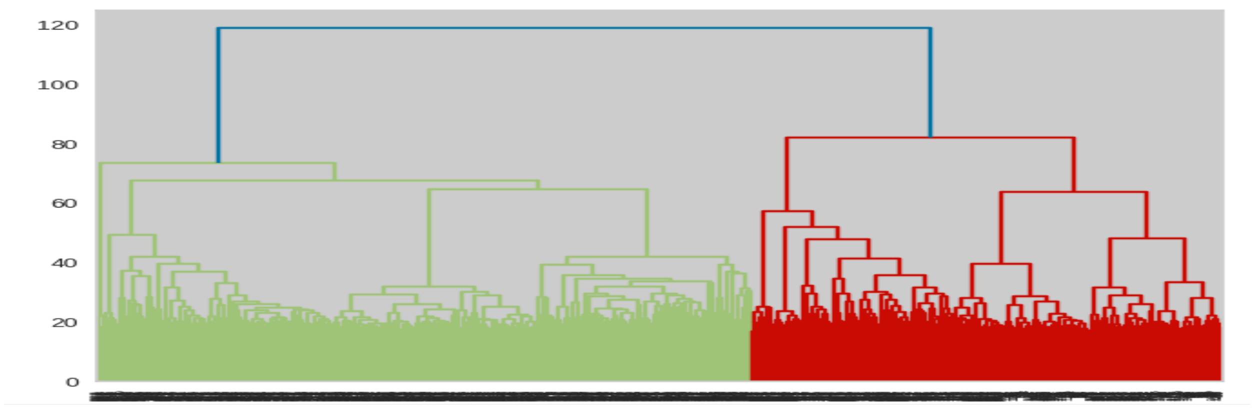
```
Hierarchical clustering with bow
```

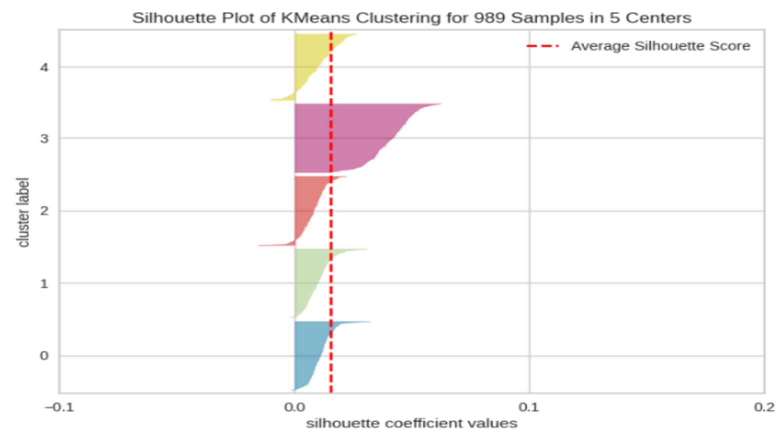


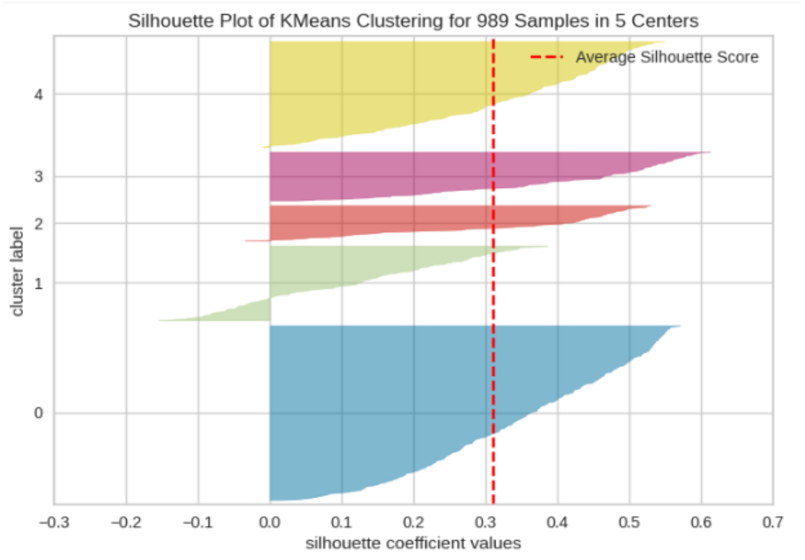
Hierarchical clustering with tfidf



Hierarchical clustering with lda

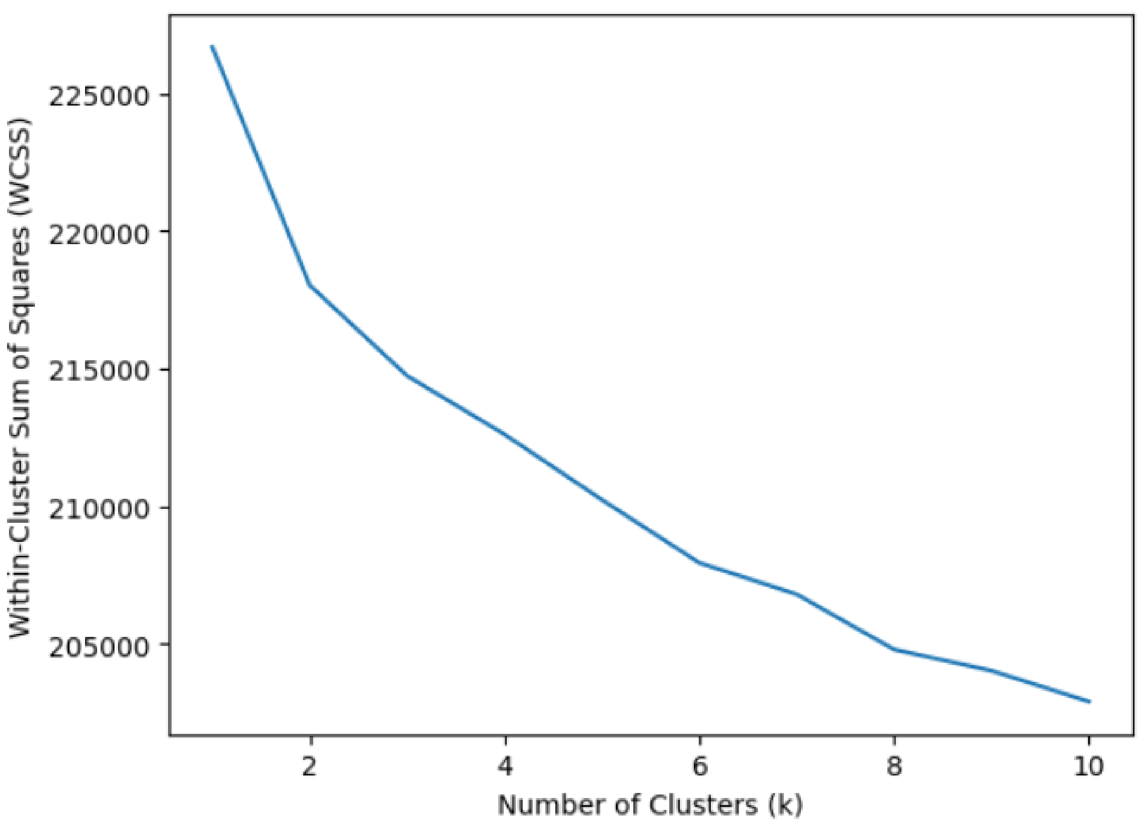






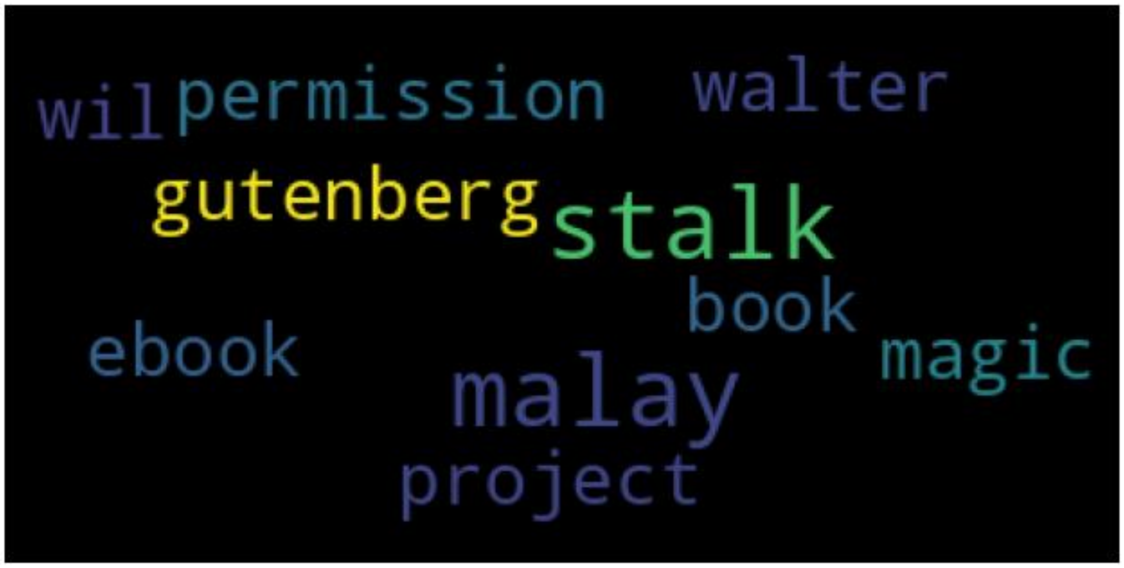
Error Analysis

The elbow method involves plotting the within-cluster sum of squares (WCSS) against the number of clusters used in K-means clustering. WCSS measures the sum of the squared distances between each data point and its assigned cluster center. As the number of clusters increases, the WCSS tends to decrease, since the data points will be closer to their assigned cluster centers. However, beyond a certain point, adding more clusters will not lead to a significant decrease in WCSS. The optimal number of clusters is typically the "elbow" point in the plot, where the rate of decrease in WCSS starts to level off.
In this code, the wcss list is initialized as an empty list, and a loop is run over the range of 1 to 10. For each value of i, a KMeans clustering model with i clusters is created, and the model is fit to the data x. The WCSS value for the resulting clustering is then appended to the wcss list.
After the loop completes, the ks list is initialized with the range of values used for the loop (1 to 10), and a line plot is created using the sns.lineplot function from the Seaborn library. The x-axis of the plot is set to ks, the y-axis is set to wcss, and the resulting plot shows the WCSS values for each number of clusters. The elbow point in the plot can be used to determine the optimal number of clusters for the data.
As as

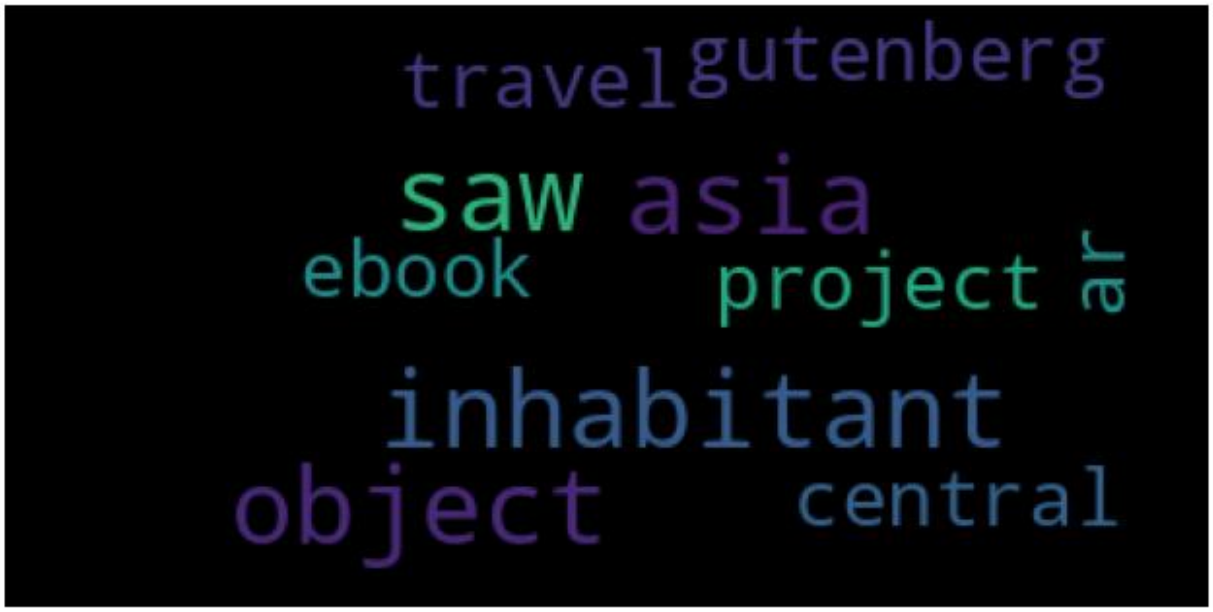


From this graph we can say that the best K is 5 and this is expected as all scores of kappa with the three algorithms is high

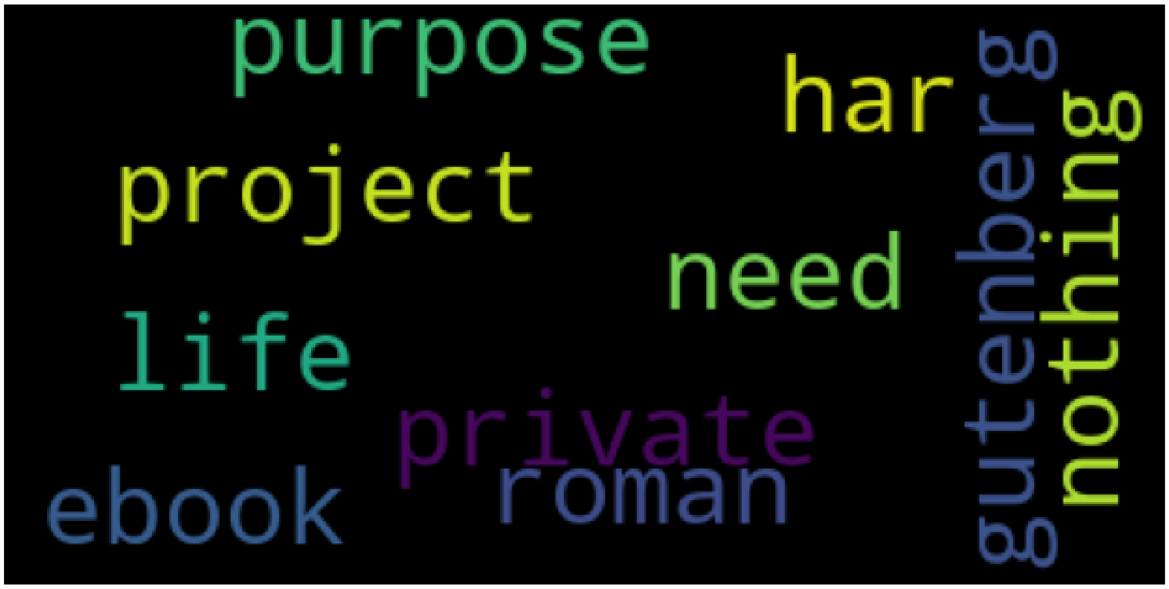
```
The most frequent 50 words of book:  Chicago by day and night
```



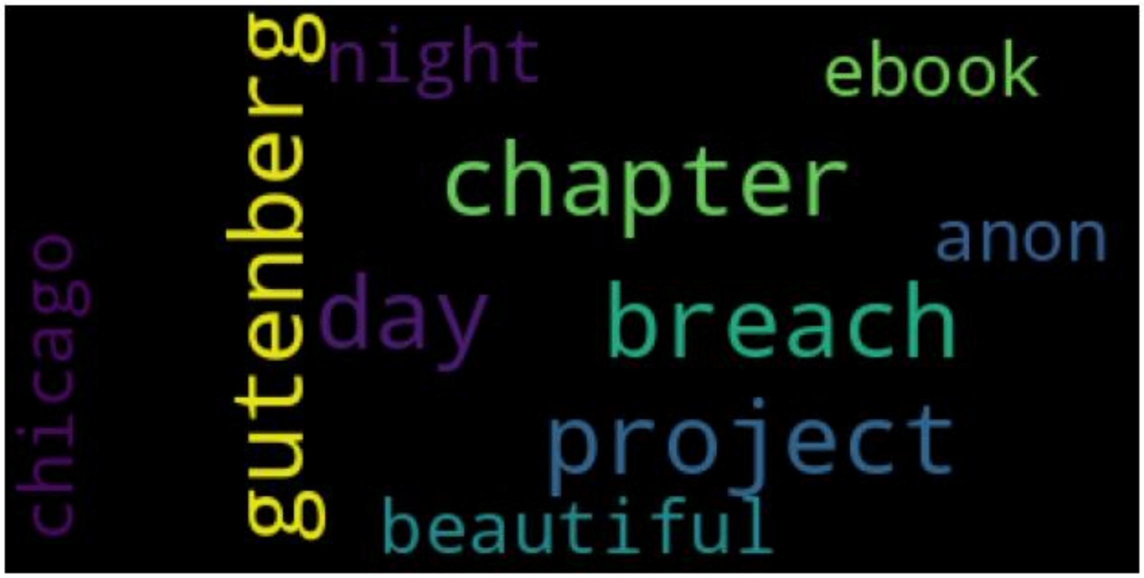
```
The most frequent 50 words of book: The Picture of Dorian Gray
```



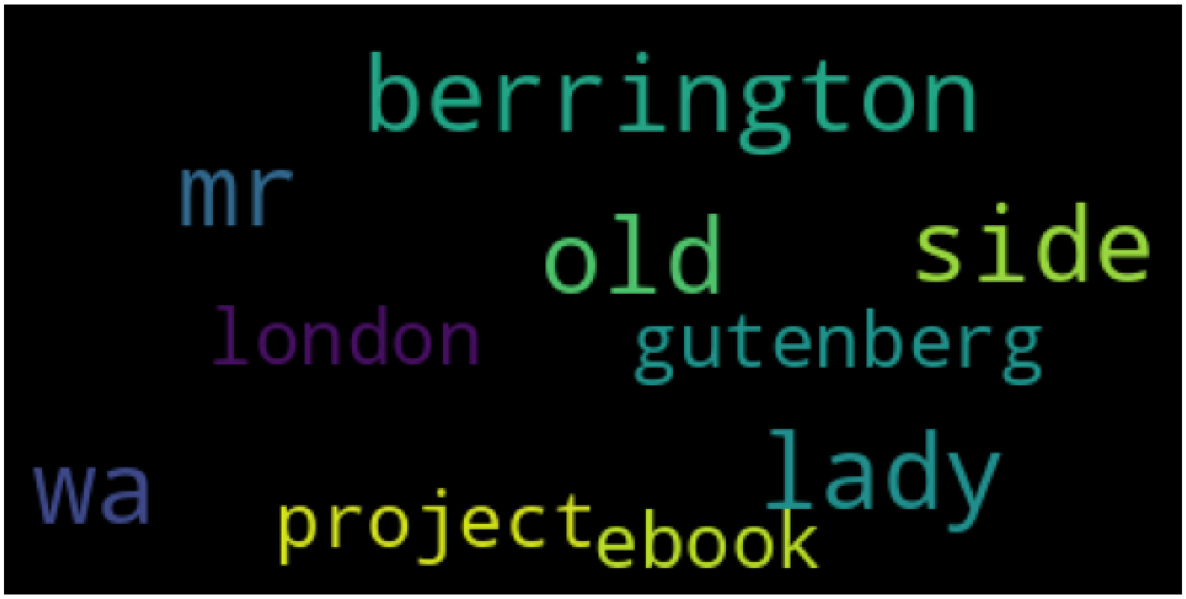
```
The most frequent 50 words of book: Malay Magic
```



```
The most frequent 50 words of book: A London Life
```



```
The most frequent 50 words of book: Travels in Central Asia
```



Programs README:

- Installation: To use this program, you will need to install the required dependencies and libraries which had been installed in the first cell.
- Customization: Some libraries might not exist in your environment like yellowbrick for example so you need to install them through command-line to download the configuration files.
- Usage: Once installed, you can run the program and specify the classification task you would like to perform.