Data Science Applications

Clustering Assignment

Abdelrahman Ibrahim

Lamis Sayed

Minah Ghanem

Mohammed El Namory

19th June, 2021

Introduction

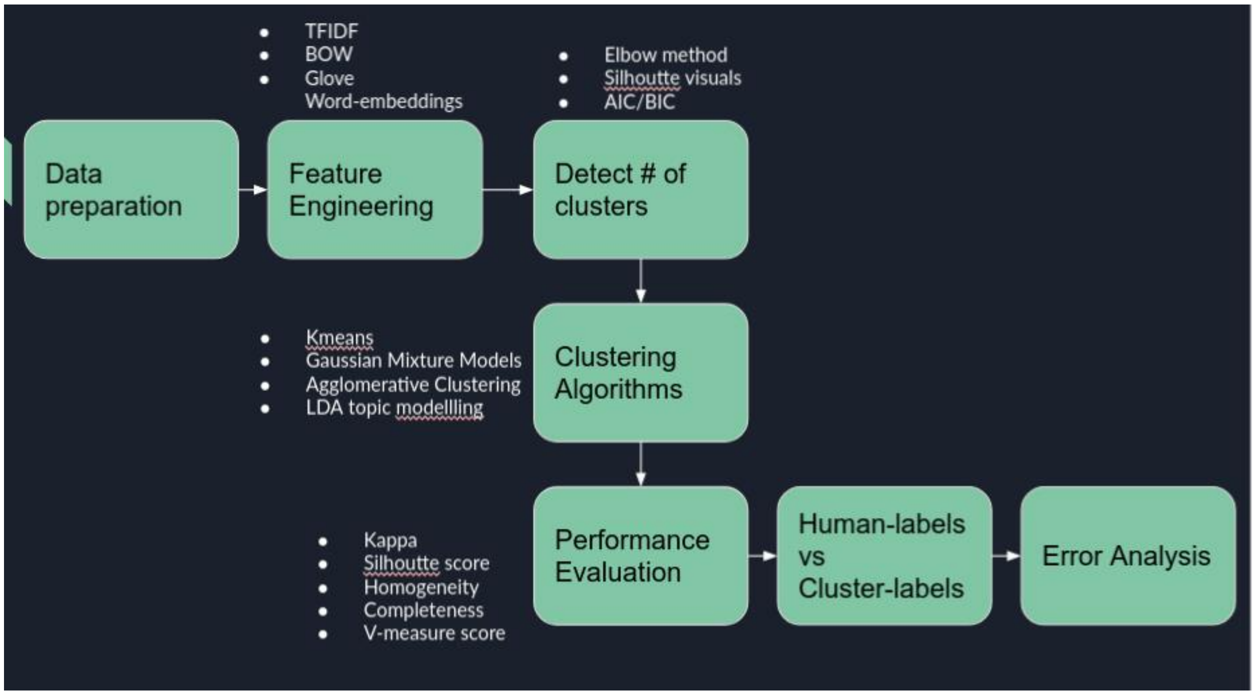
Overview

In this project, we have worked on the clustering problem on the gutenberg dataset. We chose five different books from the dataset and worked on them.

We worked through a pipeline that consists of the following steps, First by partitioning the data into 200 partitions each of length 150 for each class. Second, by trying different types of feature engineering and seeing the effect of each type on the final results.

Third, by detecting the number of clusters by using different methods and then deciding manually by the information we got. We have also tried different clustering algorithms and investigated their different performances.

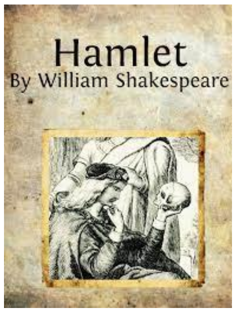
In the last step, we have tried different levels of evaluations, the first one numerically by calculating different kinds of matrices. Secondly, by comparing the human labels to the cluster labels and thirdly by going more in depth to investigate the cause of the error and in which parts the machine threw off.

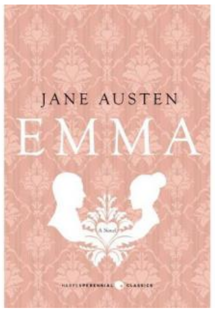


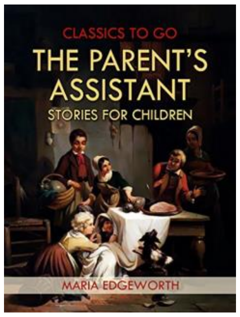
Data Chosen

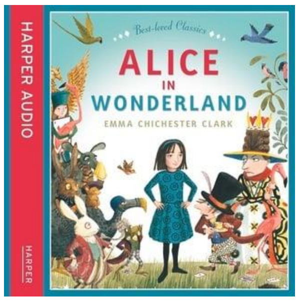
Books Selection

We have tried to choose five different books of five different genres and authors.







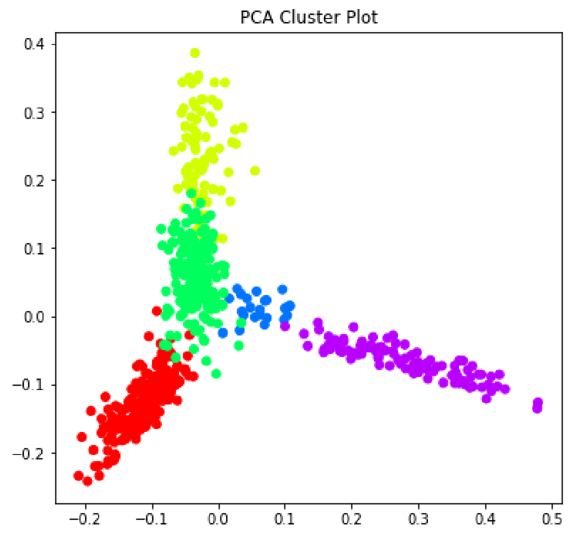


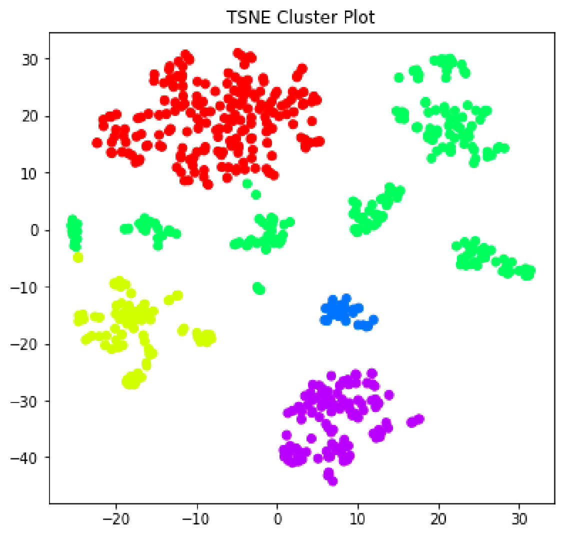
- Hamlet , **Drama by William Shakespeare**
- Emma , **Fiction by Jane Austen**
- Alice in wonderland , **Fantasy Drama by Lewis carroll and Alice Gerstenberg**
- The Parent's Assistant; Or, Stories for Children , **Children's stories by** Maria Edgeworth
- Poems of William Blake, **Poetry by William Blake**

Word Frequency distribution in each book

```
Book 0 of the author Jane Austen
[('mr', 1124), ('could', 830), ('would', 817), ('emma', 751), ('mrs', 687), ('miss', 587), ('must', 567), ('much', 474), ('said', 474), ('one', 428)]


Book 1 of the author William Blake
[('little', 45), ('thee', 42), ('like', 35), ('thou', 35), ('thy', 31), ('love', 29), ('sweet', 28), ('night', 28), ('joy', 25), ('away', 24)]


Book 2 of the author William Shakespeare
[('ham', 337), ('lord', 212), ('haue', 175), ('king', 171), ('shall', 107), ('thou', 105), ('come', 104), ('hamlet', 100), ('good', 98), ('hor', 95)]


Book 3 of the author Lewis Carroll and Alice Gerstenberg
[('said', 462), ('alice', 385), ('little', 128), ('one', 101), ('know', 86), ('like', 85), ('would', 83), ('went', 83), ('could', 77), ('thought', 74)]


Book 4 of the author Maria Edgeworth
[('said', 1427), ('would', 510), ('one', 481), ('upon', 477), ('mr', 458), ('could', 424), ('know', 416), ('little', 414), ('good', 412), ('well', 399)]
```

Plot of the Actual books

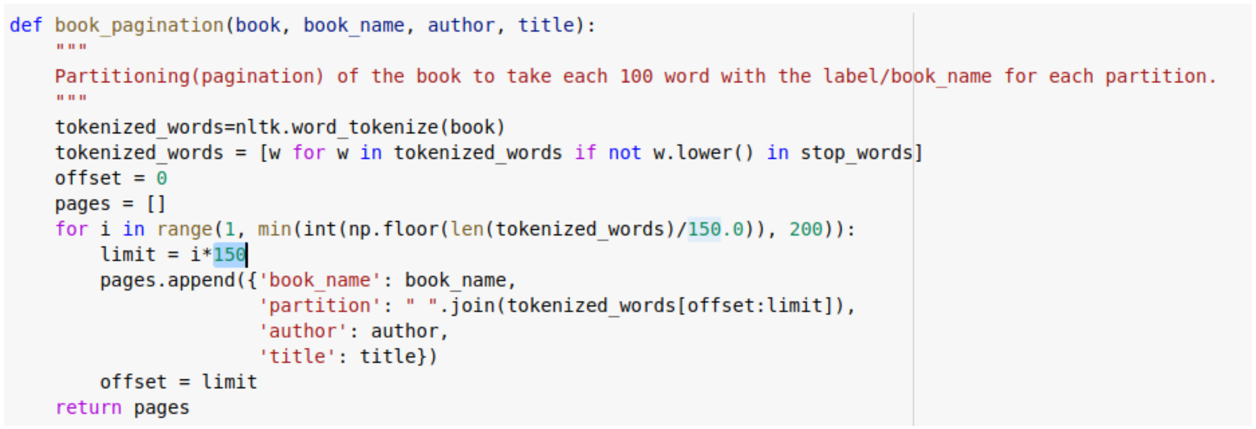




Preprocessing and Data Cleansing

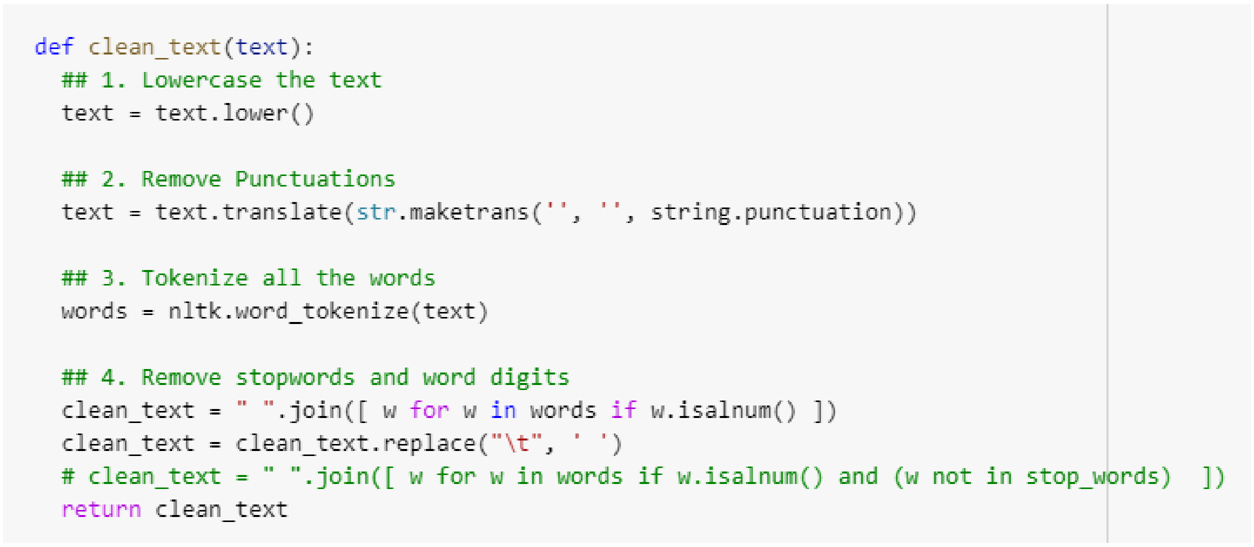
Main Functions:

Book_pagination is used to make partitions out of the text. It splits each book into **200** different partition that each consist of **150** word.



```
def book_pagination(book, book_name, author, title):
    """
    Partitioning(pagination) of the book to take each 100 word with the label/book_name for each partition.
    """
    tokenized_words=nltk.word_tokenize(book)
    tokenized_words = [w for w in tokenized_words if not w.lower() in stop_words]
    offset = 0
    pages = []
    for i in range(1, min(int(np.floor(len(tokenized_words)/150.0)), 200)):
        limit = i*150
        pages.append({'book_name': book_name,
                      'partition': " ".join(tokenized_words[offset:limit]),
                      'author': author,
                      'title': title})
        offset = limit
    return pages
```

Clean_text is used for cleaning by (lowering text - removing punctuations - keeping only alphanumeric tokens - removing any tabs - removing stop words)



```python
def clean_text(text):
  ## 1. Lowercase the text
  text = text.lower()

  ## 2. Remove Punctuations
  text = text.translate(str.maketrans('', '', string.punctuation))

  ## 3. Tokenize all the words
  words = nltk.word_tokenize(text)

  ## 4. Remove stopwords and word digits
  clean_text = " ".join([ w for w in words if w.isalnum() ])
  clean_text = clean_text.replace("\t", ' ')
  # clean_text = " ".join([ w for w in words if w.isalnum() and (w not in stop_words)  ])
  return clean_text
```

Feature Engineering

In the feature extraction phase, we have tried multiple methods and compared their results:

CountVectorizer Features:

It is a simple way of building a vocabulary of known words. The feature output of that method is an encoded vector with a length of the entire vocabulary and an integer count for the number of times each word appeared in the document.

TF-IDF Features:

In this method we have calculated the Term Frequency – Inverse Document (TF-IDF)

Which consists of **Term Frequency:** This summarizes how often a given word appears within a document. **And the Inverse Document Frequency:** This downscales words that appear a lot across documents.

It overcomes the issue of the simple count of having large counts for words that do not have meaningful insights, instead it gives more weight for words that matter.

LDA Features:

Latent Dirichlet Allocation (LDA) is an algorithm to extract topic modeling from large documents. We Made a vector out of the LDA model to use it as input for the unsupervised algorithm. The feature vector for each partition consists of a set of probabilities of all the topics. We used the soft prediction of the topic classification model as an input feature for the supervised classifier

GLoVe

GLoVe: global vector for word representation is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on aggregated global word-word co-occurrence statistics from a corpus, and the resulting representations showcase interesting linear substructures of the word vector space.

We used a pretrained model from the glove " average_word_embeddings_glove.6B.300d" which is trained on 6 billion tokens and has a feature vector of a length of 300.

K Means Algorithm

How the Algorithm Works

K-means is an iterative centroid-based algorithm, or a distance-based algorithm, where we calculate the distances to assign a point to a cluster. In K-Means, each cluster is associated with a centroid. It tries to make the intra-cluster data points as similar as possible while also keeping the clusters as different as possible.

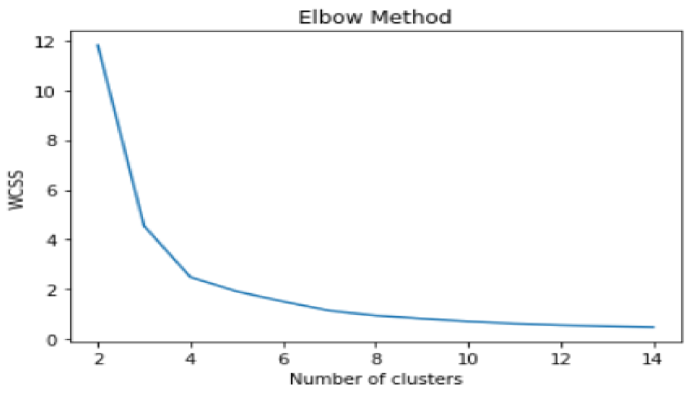
In this Algorithm we used different methods to get Features (TFIDF-BOW-Word Embeddings-LDA) , we also used a dimensionality reduction method (PCA) on these features.

Choosing Number of Clusters

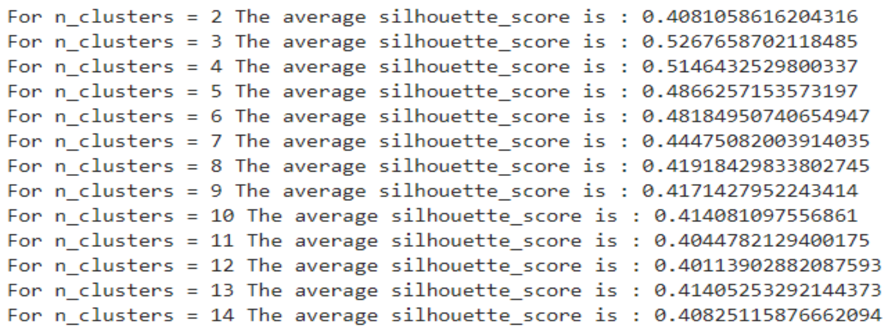
We choose two methods of determining the number of clusters: the silhouette score and the elbow method. In these two methods, we have tried different values for k (2-15) and calculated the silhouette score and the sum of squared distances to draw the elbow graph.

TFIDF : We can clearly see that there is an elbow at the number of clusters 4 also it has the highest silhouette score of 0.6.

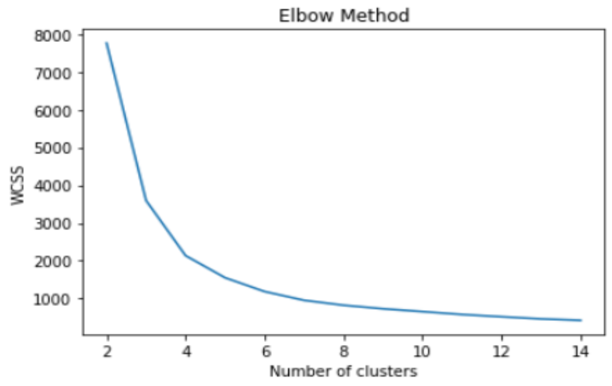
```
For n_clusters = 2 The average silhouette_score is : 0.5636698802212871
For n_clusters = 3 The average silhouette_score is : 0.5997802639991326
For n_clusters = 4 The average silhouette_score is : 0.6037857646846974
For n_clusters = 5 The average silhouette_score is : 0.5844936630289458
For n_clusters = 6 The average silhouette_score is : 0.5185202378040548
For n_clusters = 7 The average silhouette_score is : 0.47599716819445775
For n_clusters = 8 The average silhouette_score is : 0.4842630552235686
For n_clusters = 9 The average silhouette_score is : 0.47072496475880826
For n_clusters = 10 The average silhouette_score is : 0.46625884551926805
For n_clusters = 11 The average silhouette_score is : 0.4300243883197283
For n_clusters = 12 The average silhouette_score is : 0.4067811685949052
For n_clusters = 13 The average silhouette_score is : 0.4042786367265016
For n_clusters = 14 The average silhouette_score is : 0.39852174703197885
```



BOW: we got the same results applying these two methods of determining the number of clusters, k=4 where there is an elbow in the graph and a high silhouette score of 0.51



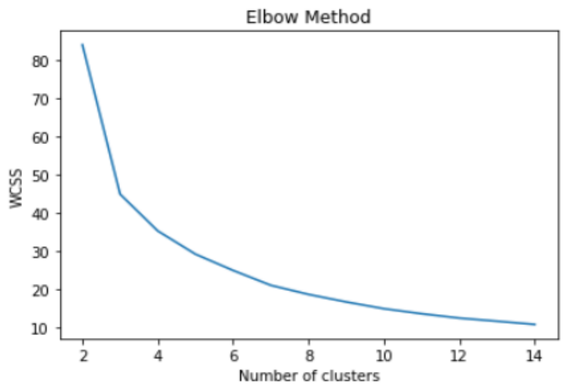
```
For n_clusters = 2 The average silhouette_score is : 0.4081058616204316
For n_clusters = 3 The average silhouette_score is : 0.5267658702118485
For n_clusters = 4 The average silhouette_score is : 0.5146432529800337
For n_clusters = 5 The average silhouette_score is : 0.4866257153573197
For n_clusters = 6 The average silhouette_score is : 0.48184950740654947
For n_clusters = 7 The average silhouette_score is : 0.44475082003914035
For n_clusters = 8 The average silhouette_score is : 0.41918429833802745
For n_clusters = 9 The average silhouette_score is : 0.4171427952243414
For n_clusters = 10 The average silhouette_score is : 0.414081097556861
For n_clusters = 11 The average silhouette_score is : 0.4044782129400175
For n_clusters = 12 The average silhouette_score is : 0.40113902882087593
For n_clusters = 13 The average silhouette_score is : 0.41405253292144373
For n_clusters = 14 The average silhouette_score is : 0.40825115876662094
```



Word Embeddings Glove:

This time the elbow is at k=3 with a silhouette score of 0.49

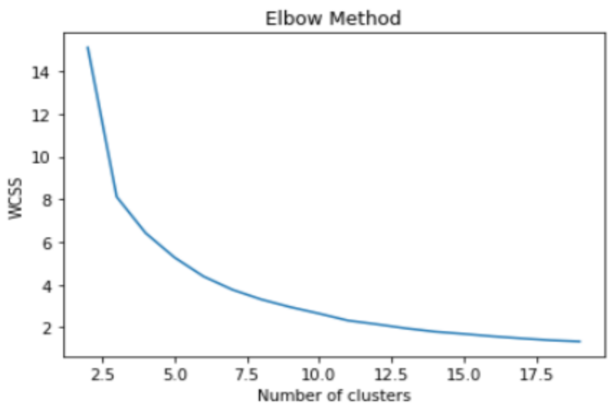
```
For n_clusters = 2 The average silhouette_score is : 0.604696
For n_clusters = 3 The average silhouette_score is : 0.49197975
For n_clusters = 4 The average silhouette_score is : 0.45046747
For n_clusters = 5 The average silhouette_score is : 0.38390592
For n_clusters = 6 The average silhouette_score is : 0.39759043
For n_clusters = 7 The average silhouette_score is : 0.36130068
For n_clusters = 8 The average silhouette_score is : 0.35356638
For n_clusters = 9 The average silhouette_score is : 0.34769538
For n_clusters = 10 The average silhouette_score is : 0.3502234
For n_clusters = 11 The average silhouette_score is : 0.35525978
For n_clusters = 12 The average silhouette_score is : 0.35052758
For n_clusters = 13 The average silhouette_score is : 0.33854643
For n_clusters = 14 The average silhouette_score is : 0.34778282
```



LDA:

Same as the previous time, the elbow is at k=3 with a silhouette score 0f 0.48

```
For n_clusters = 2 The average silhouette_score is : 0.59654266
For n_clusters = 3 The average silhouette_score is : 0.4846462
For n_clusters = 4 The average silhouette_score is : 0.419097
For n_clusters = 5 The average silhouette_score is : 0.41661885
For n_clusters = 6 The average silhouette_score is : 0.39479074
For n_clusters = 7 The average silhouette_score is : 0.378408
For n_clusters = 8 The average silhouette_score is : 0.38687485
For n_clusters = 9 The average silhouette_score is : 0.363436
For n_clusters = 10 The average silhouette_score is : 0.35294938
For n_clusters = 11 The average silhouette_score is : 0.3639313
For n_clusters = 12 The average silhouette_score is : 0.35619617
For n_clusters = 13 The average silhouette_score is : 0.36170867
For n_clusters = 14 The average silhouette_score is : 0.37176546
For n_clusters = 15 The average silhouette_score is : 0.3606562
For n_clusters = 16 The average silhouette_score is : 0.36040717
For n_clusters = 17 The average silhouette_score is : 0.36116958
For n_clusters = 18 The average silhouette_score is : 0.36058602
For n_clusters = 19 The average silhouette_score is : 0.35732856
```

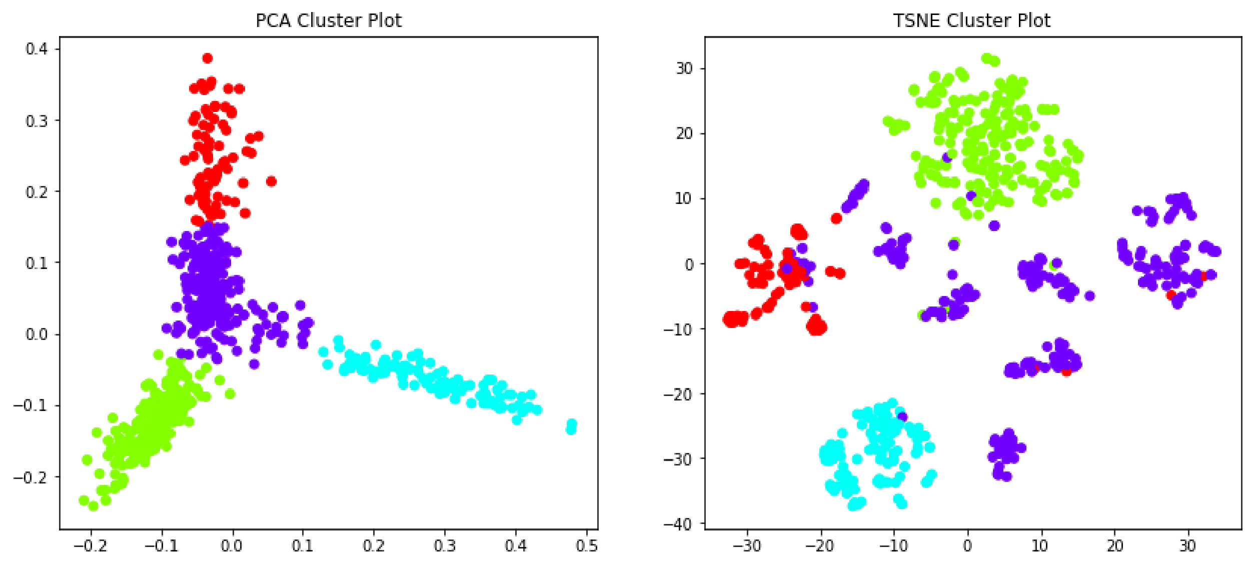


- **Based on these results, we choose k=4 for tf-idf and bow features and k=3 for LDA and Glove.**
- **In all the silhouette scores, we can clearly see that the score is pretty high when the k=2, due to that Shakespeare has a unique writing style that is different from the other 4 books.**

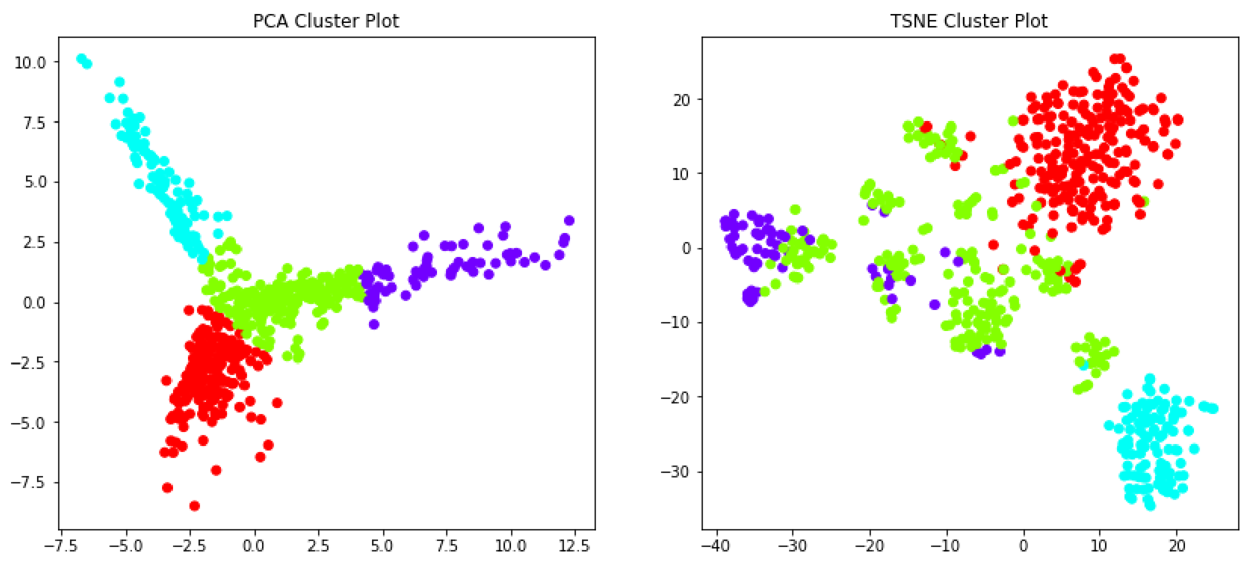
Visualizing the Clustering Results

To visualize the clustering results, we used PCA and TSNE for plotting the different clusters ang got the following result.

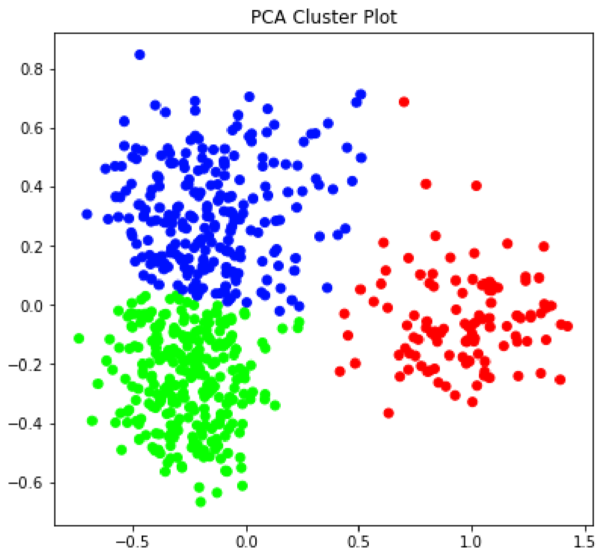
TFIDF

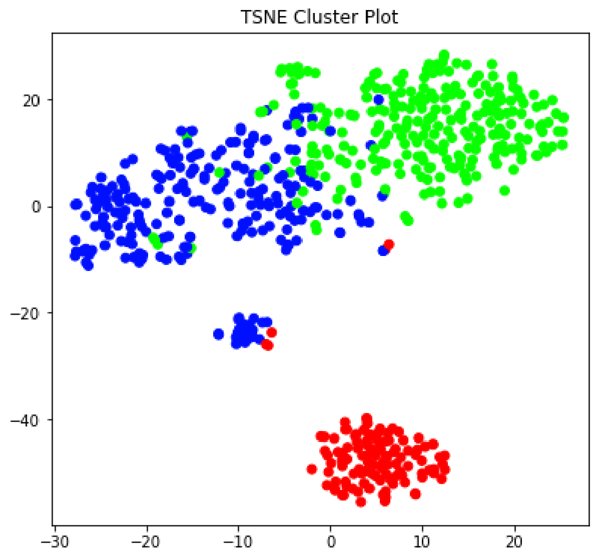

PCA Cluster Plot
TSNE Cluster Plot

BOW

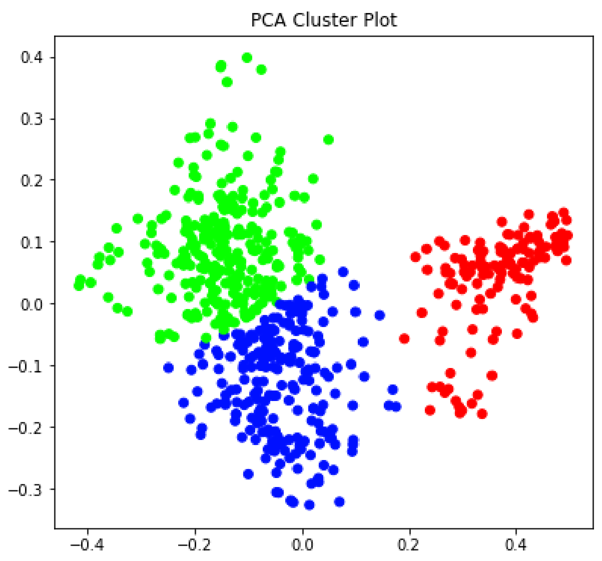

PCA Cluster Plot
TSNE Cluster Plot

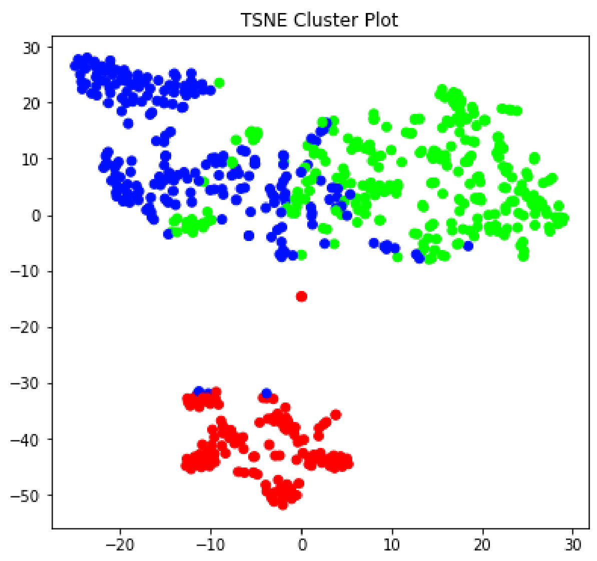
Glove


PCA Cluster Plot
0.8
0.6
0.4
0.2
0.0
−0.2
−0.4
−0.6
−0.5
0.0
0.5
1.0
1.5


TSNE Cluster Plot
20
0
−20
−40
−30
−20
−10
0
10
20

LDA





There is always one class that is clearly separable than the other classes which confirms that Shakespeare has a unique way of writing.

Perform Evaluations=

To Evaluate our clustering, we choose multiple metrics for evaluation:

Homogeneity Score: This Score should be high if cluster contains only samples belonging to a single class

Completeness Score: This score should be high if if all the data points that are members of a given class are elements of the same cluster

V-measure Score: The harmonic mean of homogeneity and completeness

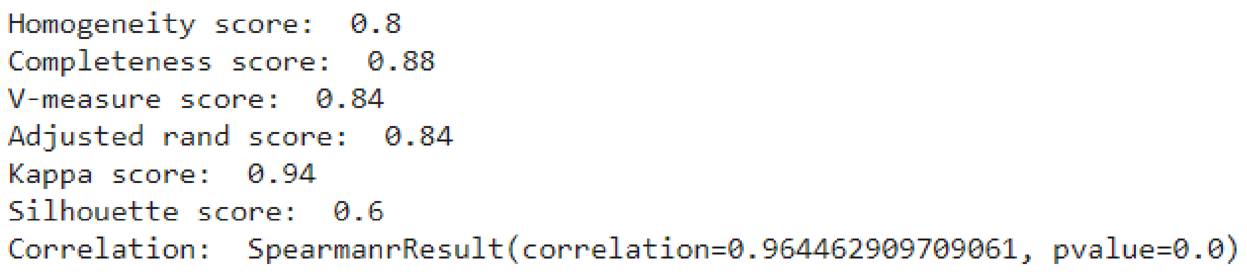
Adjusted rand Score: The Rand Index computes a similarity measure between two clusterings by considering all pairs

Kappa Score:A statistic that measures inter-annotator agreement

Silhouette Score: A method of interpretation and validation of consistency within clusters of data

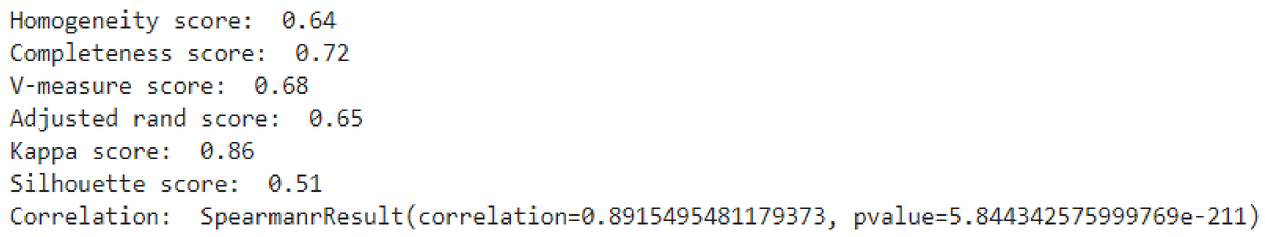
SearmanrResult: Spearman correlation coefficient with associated p-value

TFIDF Results:



```
Homogeneity score:  0.8
Completeness score:  0.88
V-measure score:  0.84
Adjusted rand score:  0.84
Kappa score:  0.94
Silhouette score:  0.6
Correlation:  SpearmanrResult(correlation=0.964462909709061, pvalue=0.0)
```

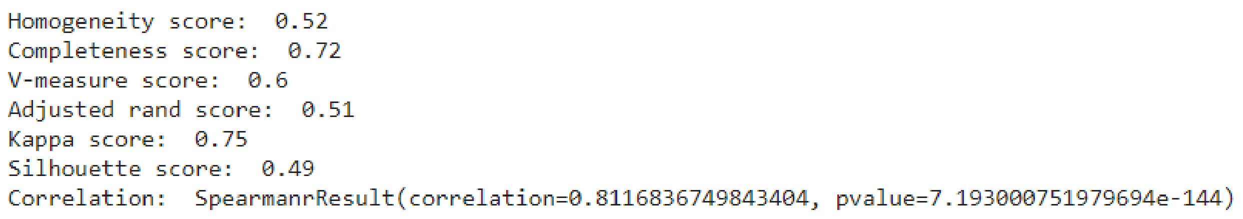
BOW Results



```
Homogeneity score:  0.64
Completeness score:  0.72
V-measure score:  0.68
Adjusted rand score:  0.65
Kappa score:  0.86
Silhouette score:  0.51
Correlation:  SpearmanrResult(correlation=0.8915495481179373, pvalue=5.844342575999769e-211)
```

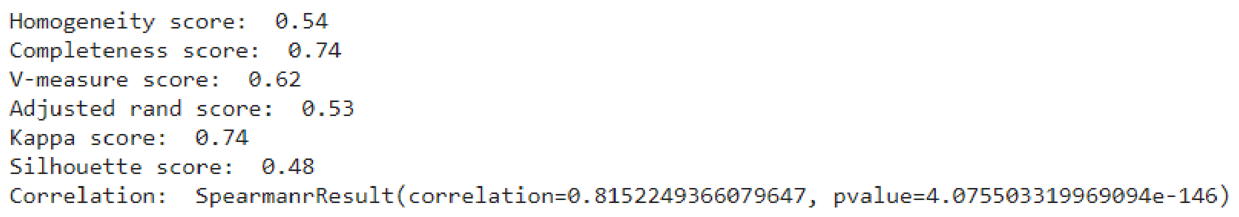
The TFIDF performs better than the bow although they run on the same number of clusters, In terms of the homogeneity, completeness, kappa and silhouette scores.

Glove Results



```
Homogeneity score:  0.52
Completeness score:  0.72
V-measure score:  0.6
Adjusted rand score:  0.51
Kappa score:  0.75
Silhouette score:  0.49
Correlation:  SpearmanrResult(correlation=0.8116836749843404, pvalue=7.193000751979694e-144)
```

LDA Results



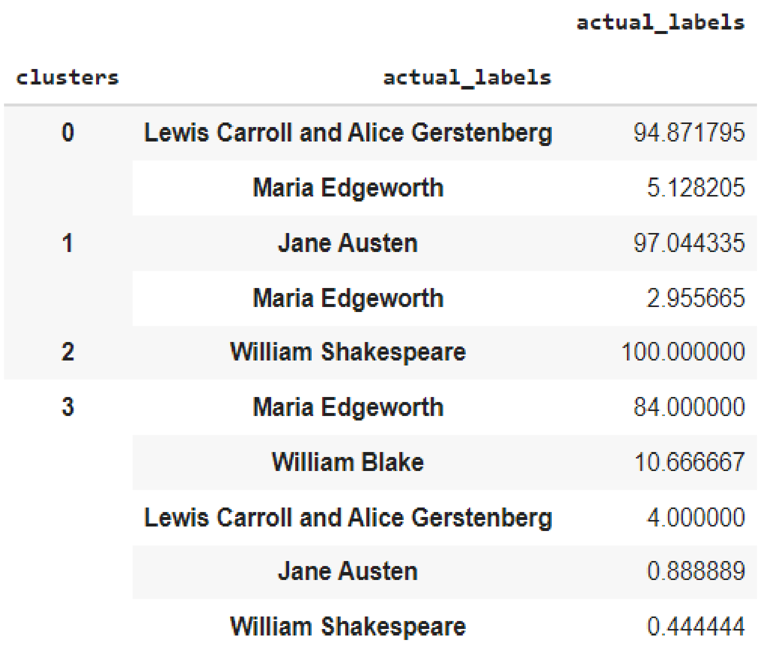
```
Homogeneity score:  0.54
Completeness score:  0.74
V-measure score:  0.62
Adjusted rand score:  0.53
Kappa score:  0.74
Silhouette score:  0.48
Correlation:  SpearmanrResult(correlation=0.8152249366079647, pvalue=4.075503319969094e-146)
```

Regarding LDA and Glove, they both run on 3 clusters, and the results are quite similar to each other. Although LDA performed a little bit better than Glove in terms of completeness and homogeneity, Glove performed better in terms of kappa and silhouette score.

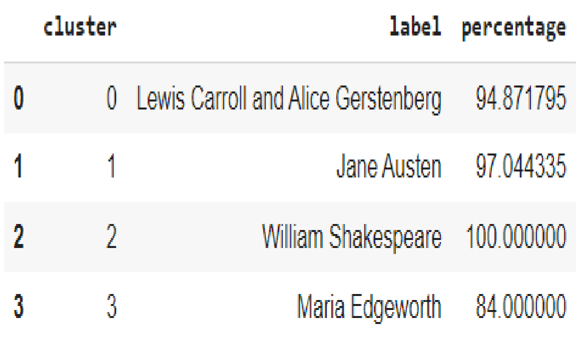
Human Labels Vs Predicted Labels

In order to compare the predicted labels to the human labels we wrote a function to calculate the percentage of classes in each cluster and then choose a certain threshold to choose the class of each cluster. The class with the highest percentage of occurrences in the cluster and exceeds the threshold is said to be the actual label of the cluster

TFIDF



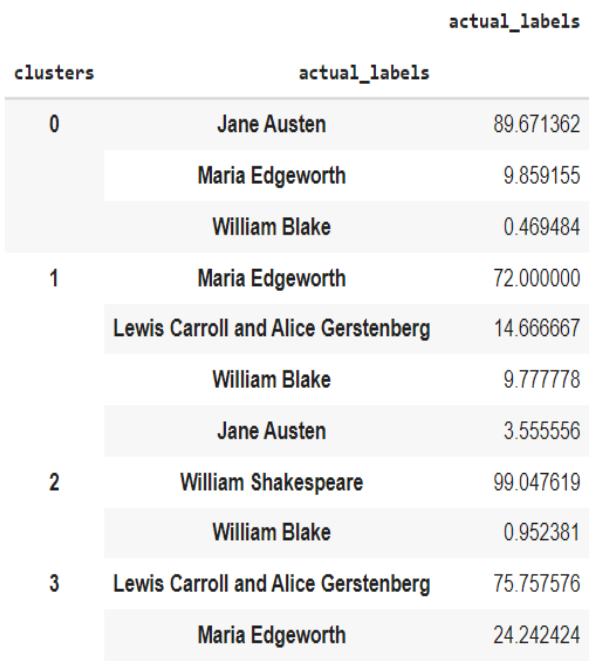
		actual_labels
clusters	actual_labels	
0	Lewis Carroll and Alice Gerstenberg	94.871795
	Maria Edgeworth	5.128205
1	Jane Austen	97.044335
	Maria Edgeworth	2.955665
2	William Shakespeare	100.000000
3	Maria Edgeworth	84.000000
	William Blake	10.666667
	Lewis Carroll and Alice Gerstenberg	4.000000
	Jane Austen	0.888889
	William Shakespeare	0.444444



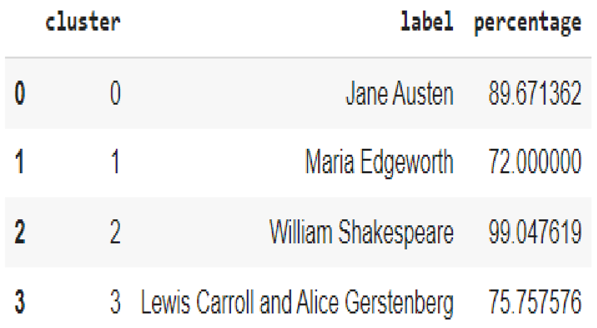
	cluster	label	percentage
0	0	Lewis Carroll and Alice Gerstenberg	94.871795
1	1	Jane Austen	97.044335
2	2	William Shakespeare	100.000000
3	3	Maria Edgeworth	84.000000

- We can clearly see that in each cluster there is a dominant class.
- Shakespeare was the one that clustered perfectly.
- The last cluster contained a lot of classes, which needed to be clustered more.

BOW



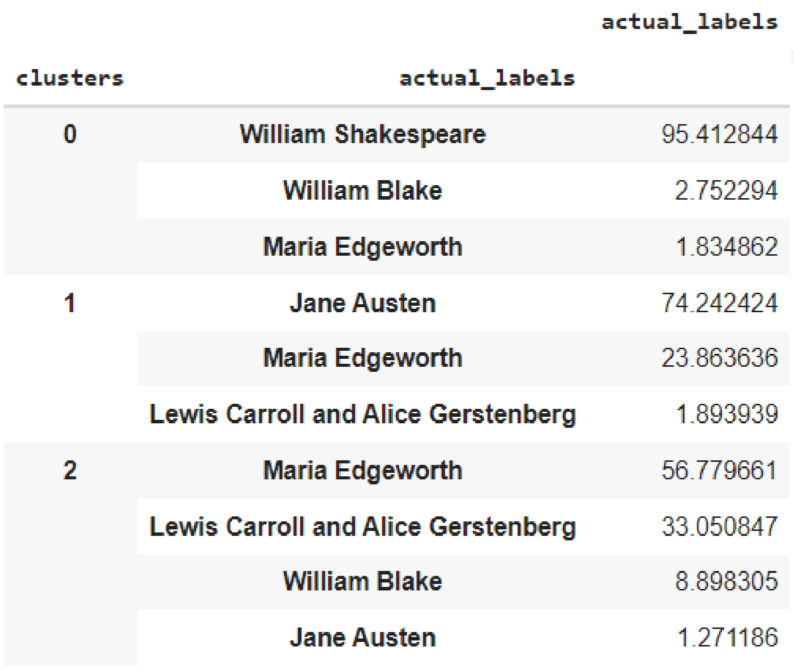
		actual_labels
clusters	**actual_labels**	
0	**Jane Austen**	89.671362
	Maria Edgeworth	9.859155
	William Blake	0.469484
1	**Maria Edgeworth**	72.000000
	Lewis Carroll and Alice Gerstenberg	14.666667
	William Blake	9.777778
	Jane Austen	3.555556
2	**William Shakespeare**	99.047619
	William Blake	0.952381
3	**Lewis Carroll and Alice Gerstenberg**	75.757576
	Maria Edgeworth	24.242424



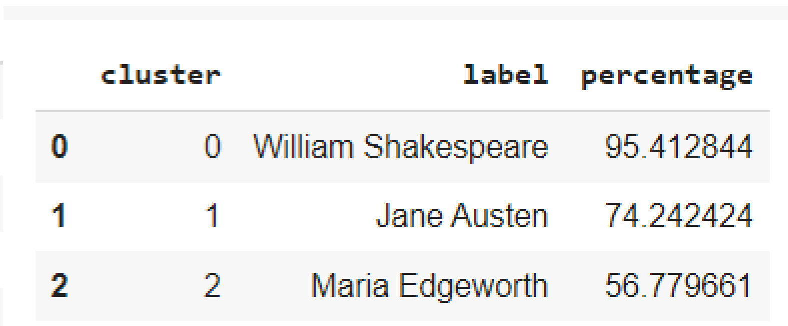
	cluster	label	percentage
0	0	Jane Austen	89.671362
1	1	Maria Edgeworth	72.000000
2	2	William Shakespeare	99.047619
3	3	Lewis Carroll and Alice Gerstenberg	75.757576

- We can clearly see that in each cluster there is a dominant class.
- Shakespeare was the one that clustered with the most confidence level.
- William Blake with distributed among the classes and it was hard to tell that it belonged to a certain class

Glove

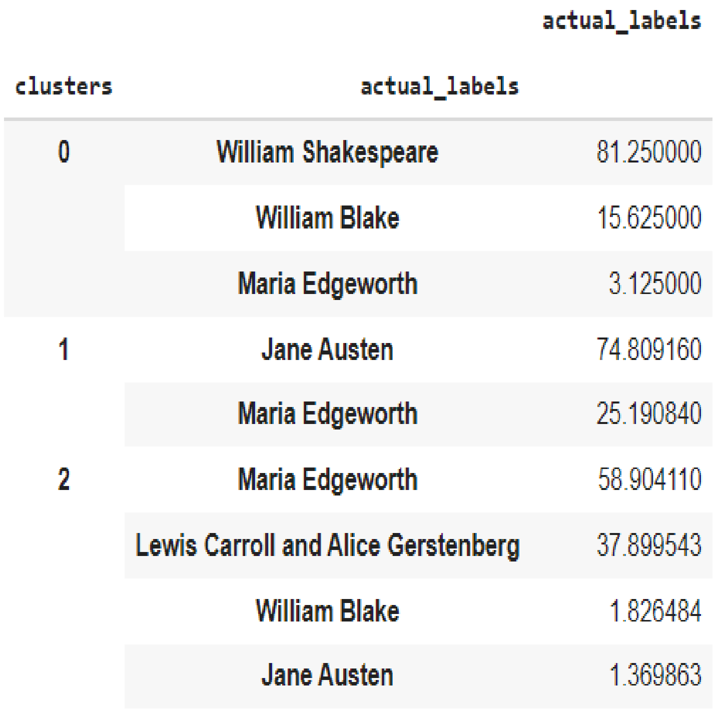


		actual_labels
clusters	actual_labels	
0	William Shakespeare	95.412844
	William Blake	2.752294
	Maria Edgeworth	1.834862
1	Jane Austen	74.242424
	Maria Edgeworth	23.863636
	Lewis Carroll and Alice Gerstenberg	1.893939
2	Maria Edgeworth	56.779661
	Lewis Carroll and Alice Gerstenberg	33.050847
	William Blake	8.898305
	Jane Austen	1.271186

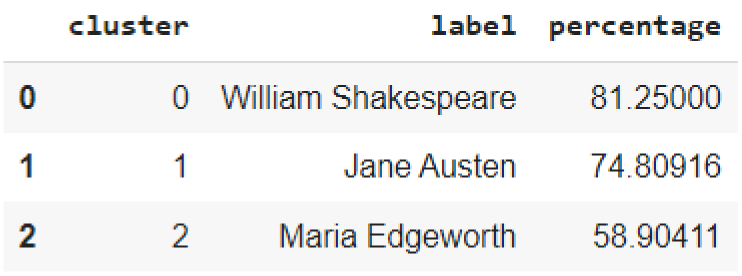


	cluster	label	percentage
0	0	William Shakespeare	95.412844
1	1	Jane Austen	74.242424
2	2	Maria Edgeworth	56.779661

LDA



		actual_labels
clusters	actual_labels	
0	William Shakespeare	81.250000
	William Blake	15.625000
	Maria Edgeworth	3.125000
1	Jane Austen	74.809160
	Maria Edgeworth	25.190840
2	Maria Edgeworth	58.904110
	Lewis Carroll and Alice Gerstenberg	37.899543
	William Blake	1.826484
	Jane Austen	1.369863



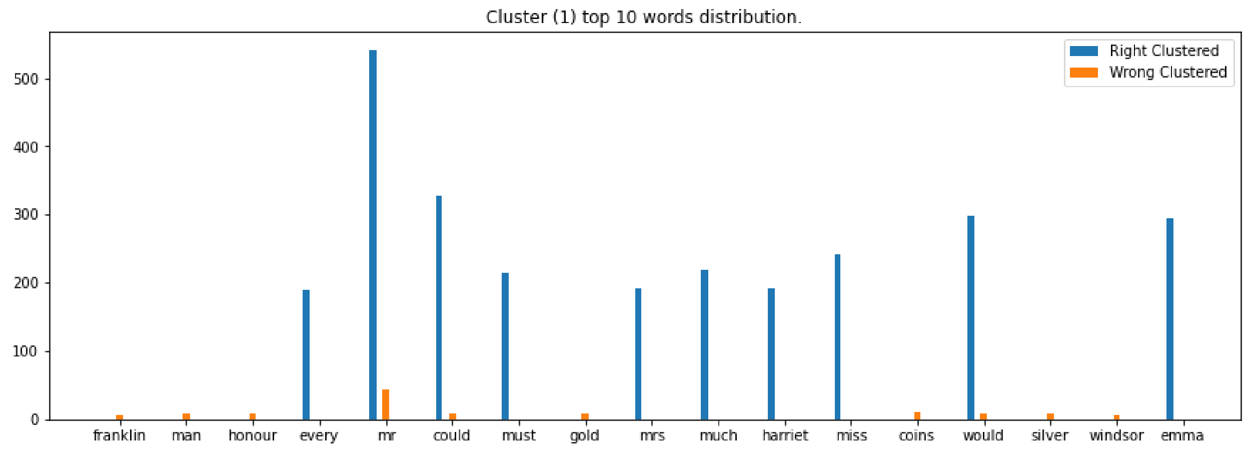
	cluster	label	percentage
0	0	William Shakespeare	81.25000
1	1	Jane Austen	74.80916
2	2	Maria Edgeworth	58.90411

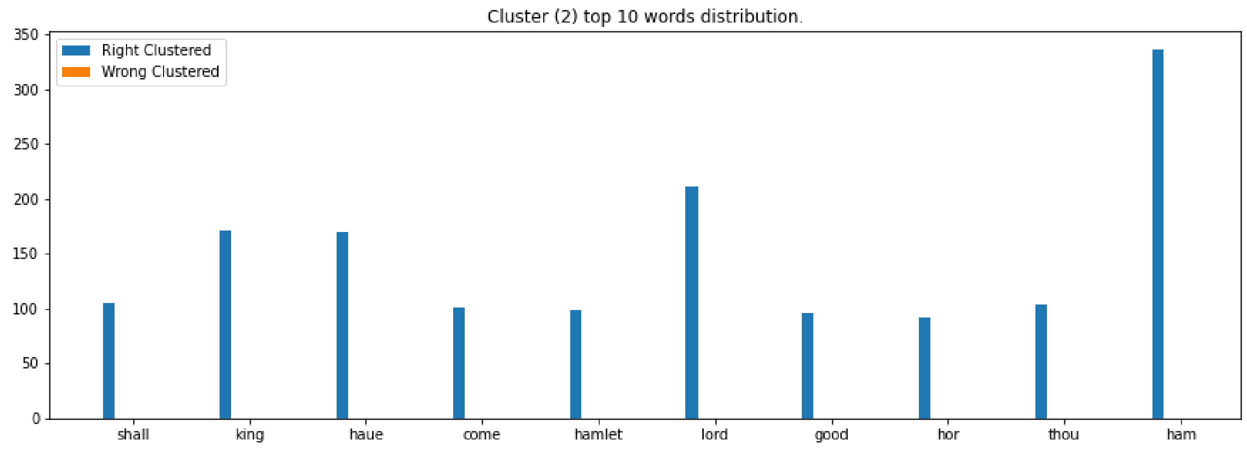
- Due to the decreasing number of clusters in both the GLOVE and the LDA, the confidence level is decreased.
- The two classes that were always clustered with the highest confidence are Shakesaper and Jane Austen

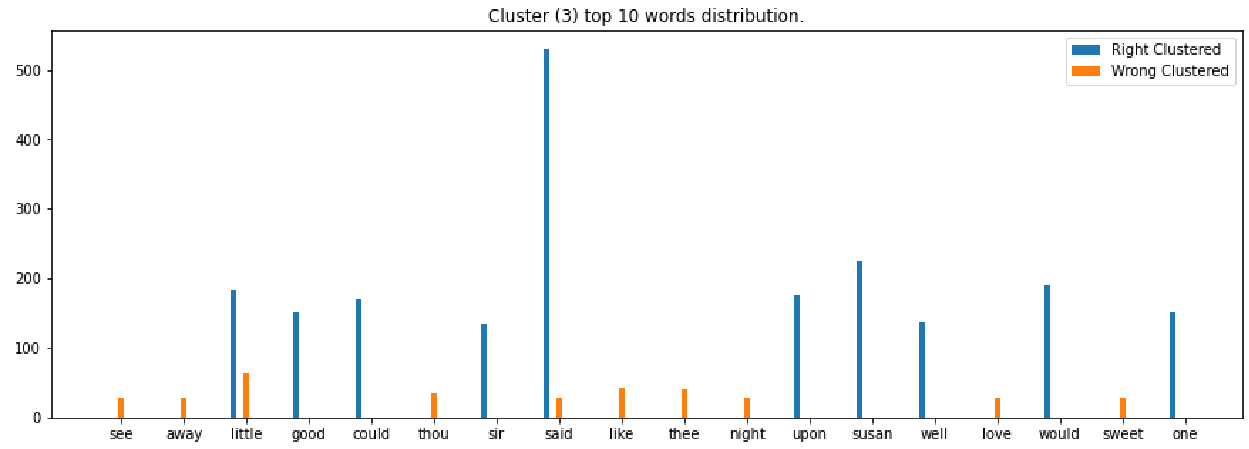
Perform Error Analysis

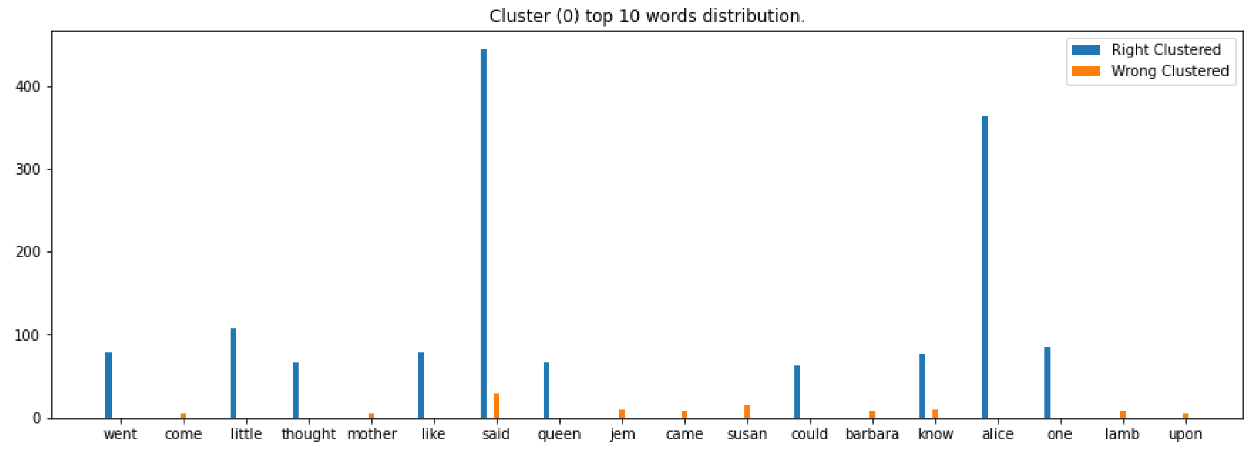
To investigate more in depth, the parts that threw the machine off and caused the error, we have plotted the 10 most words that were found and each cluster and whether or not they were mis-clustterd.

TFIDF

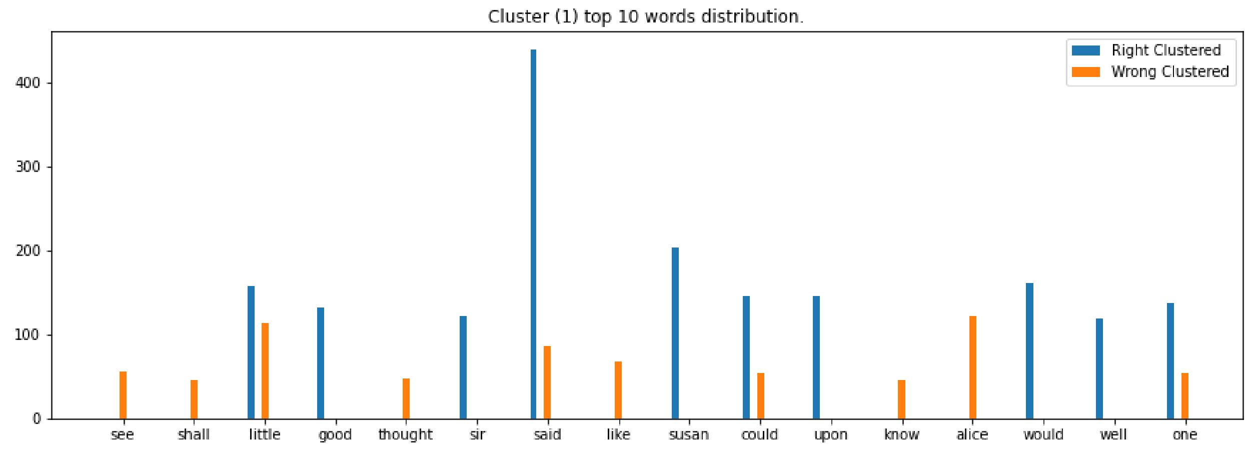


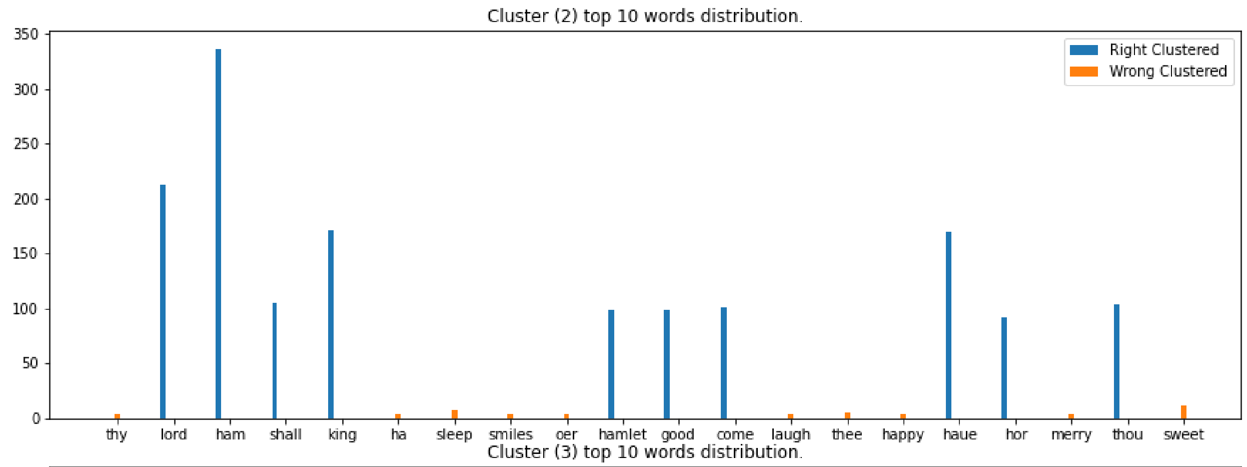


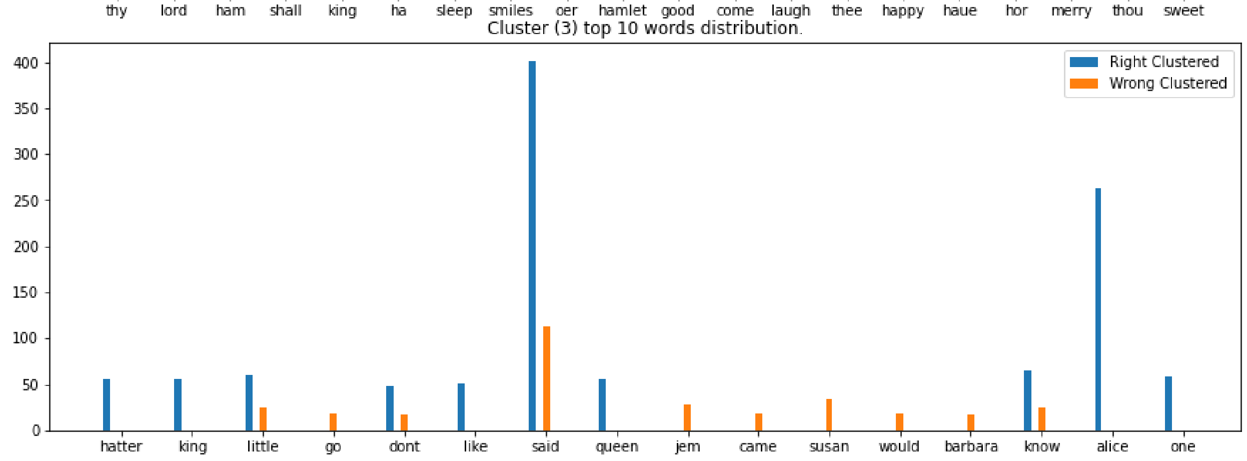

Cluster (3) top 10 words distribution.
Right Clustered
Wrong Clustered
0
100
200
300
400
500
see
away
little
good
could
thou
sir
said
like
thee
night
upon
susan
well
love
would
sweet
one

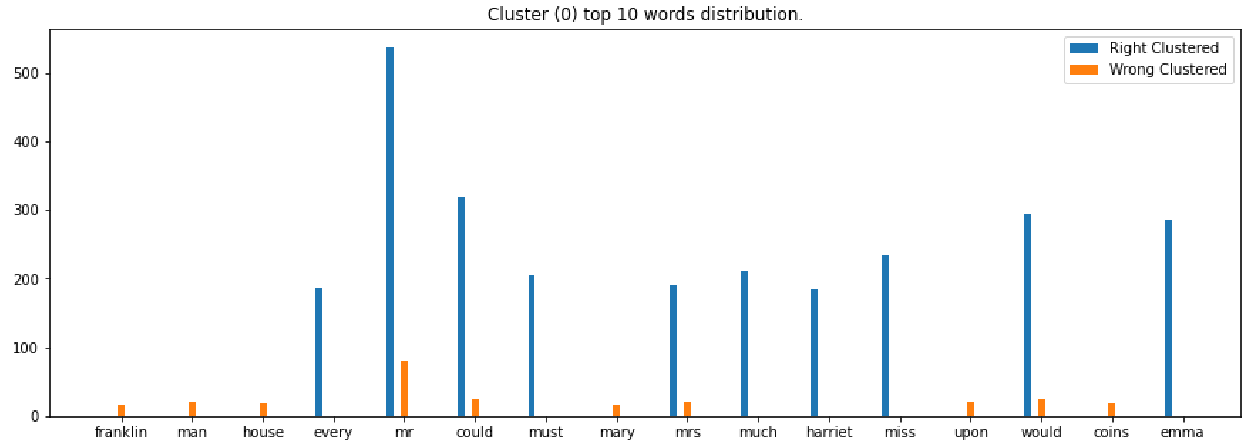

Cluster (0) top 10 words distribution.
Right Clustered
Wrong Clustered
0
100
200
300
400
went
come
little
thought
mother
like
said
queen
jem
came
susan
could
barbara
know
alice
one
lamb
upon

BOW


Cluster (1) top 10 words distribution.
Right Clustered
Wrong Clustered
0
100
200
300
400
see
shall
little
good
thought
sir
said
like
susan
could
upon
know
alice
would
well
one

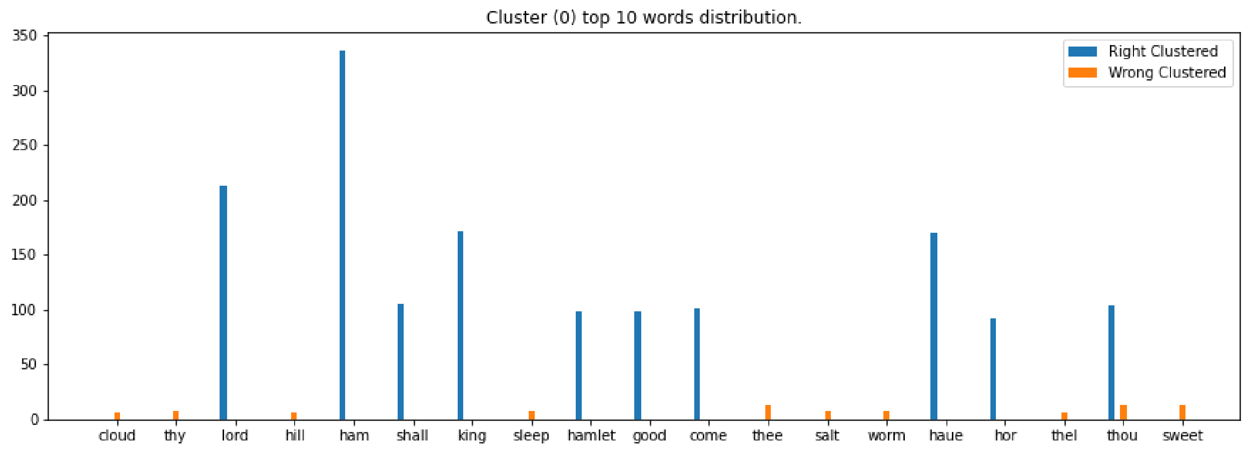

Cluster (2) top 10 words distribution.
Right Clustered
Wrong Clustered
thy
lord
ham
shall
king
ha
sleep
smiles
oer
hamlet
good
come
laugh
thee
happy
haue
hor
merry
thou
sweet

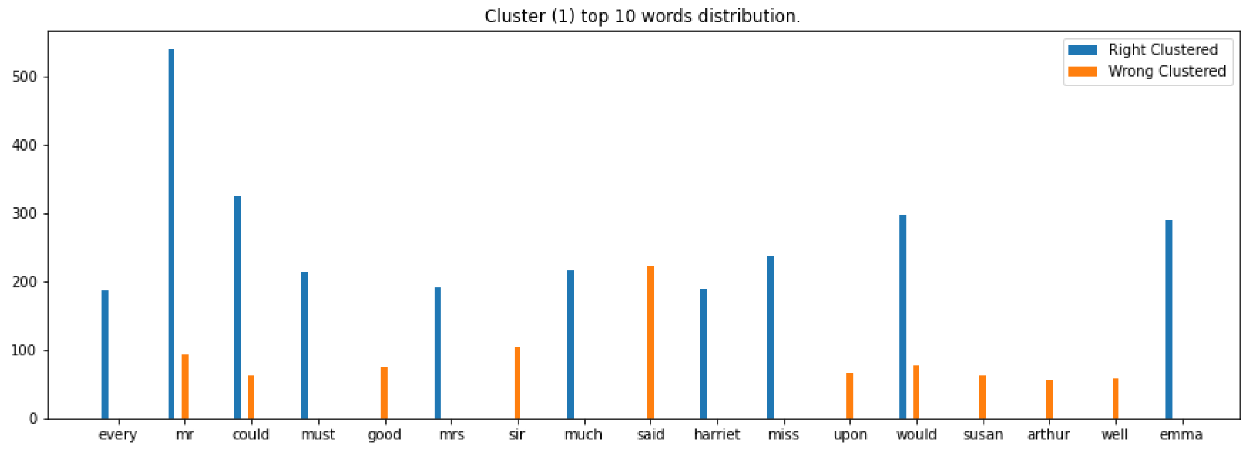

Cluster (3) top 10 words distribution.
Right Clustered
Wrong Clustered
hatter
king
little
go
dont
like
said
queen
jem
came
susan
would
barbara
know
alice
one

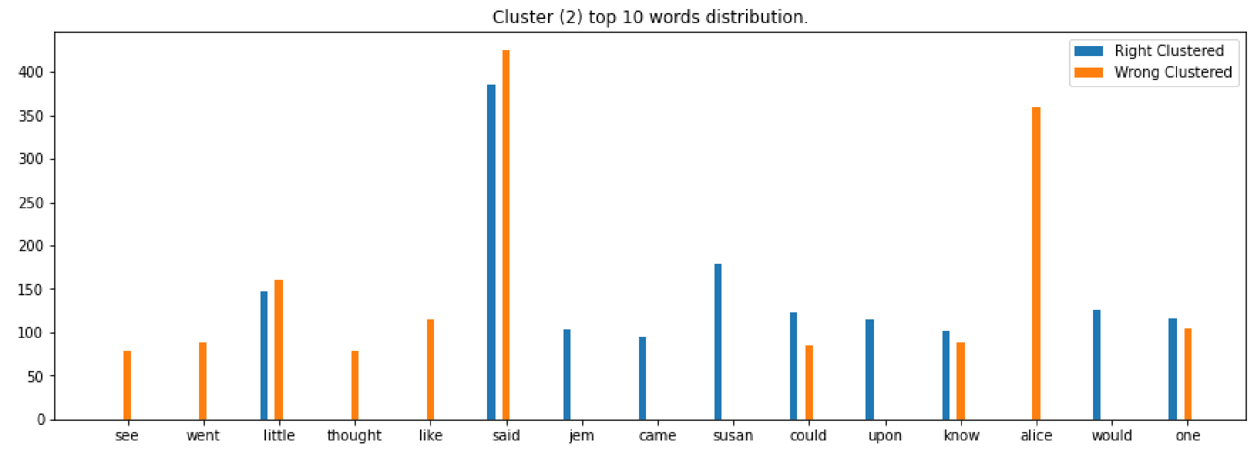

Cluster (0) top 10 words distribution.
Right Clustered
Wrong Clustered
franklin
man
house
every
mr
could
must
mary
mrs
much
harriet
miss
upon
would
coins
emma

- We can see that there is more intersection in the bow methods as it only calculates the counts of the word in the corpus.
- The intersection is obvious in the bow method in cluster number 1 in the word “little” and the word “said”. “Said” have also some intersection in the cluster number 3

Glove

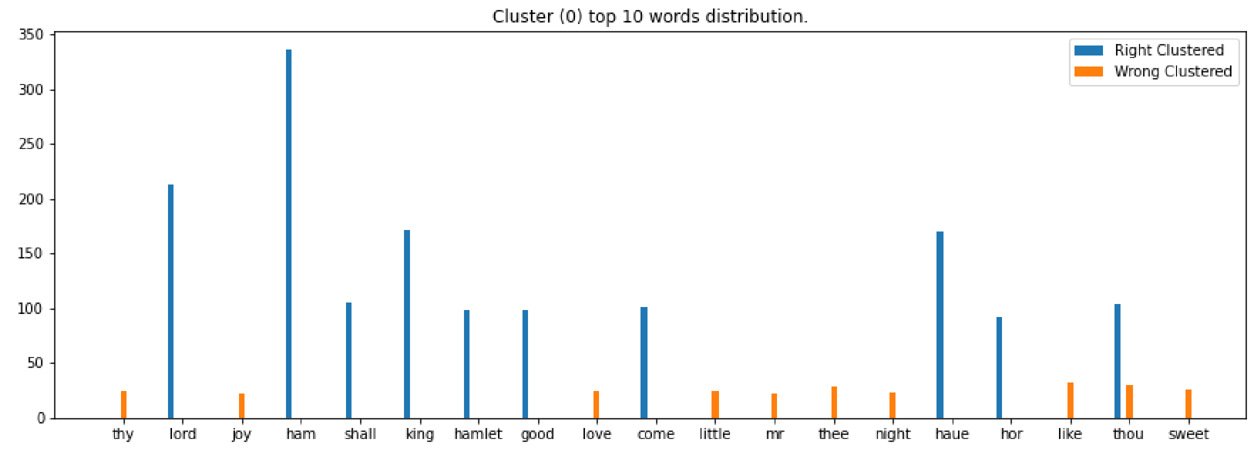


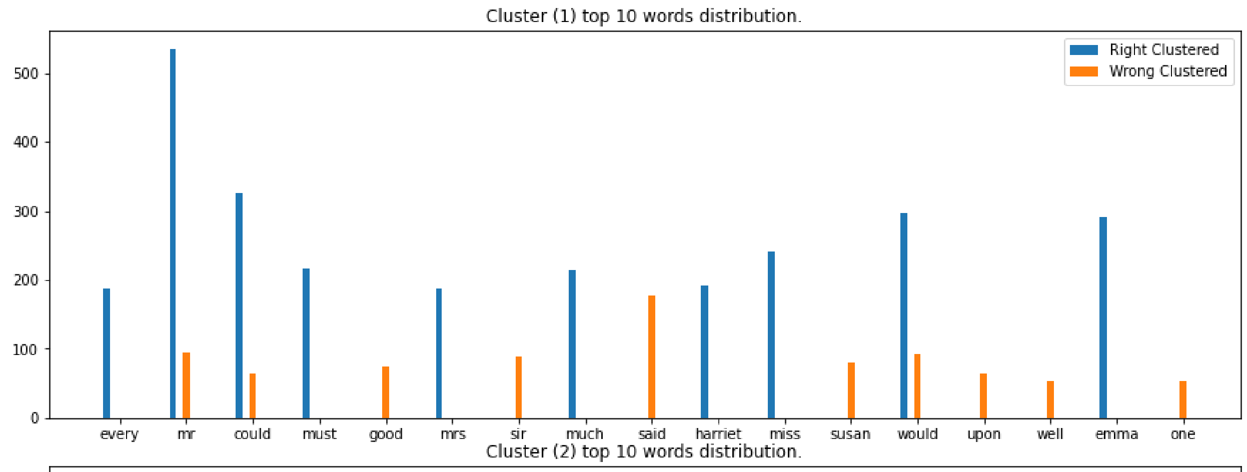


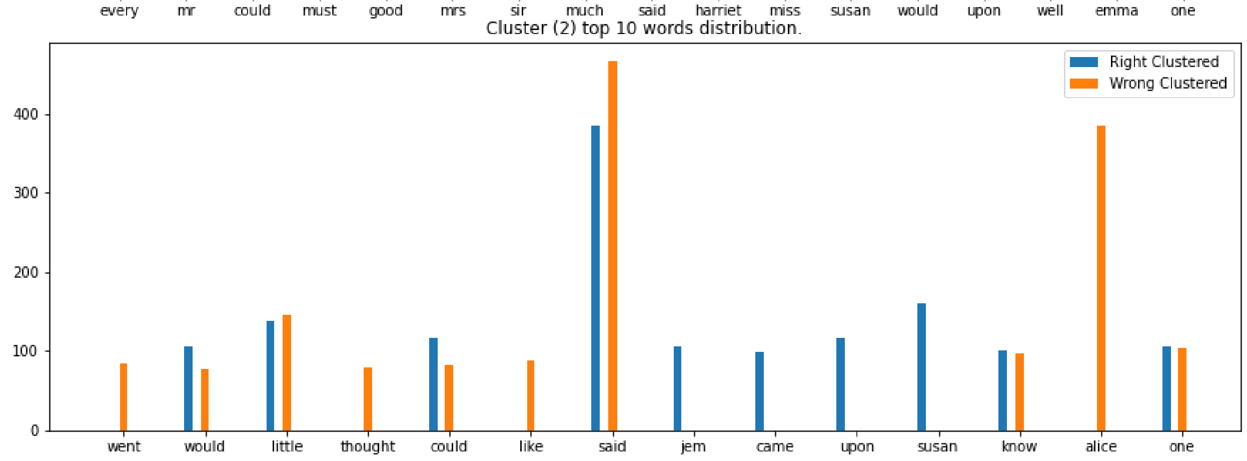


- **Cluster 2 has the most error and intersections between words that are miss-clustered and clustered correctly. Words like “said” that are common in literature confused the machine and made it throw off.**

LDA



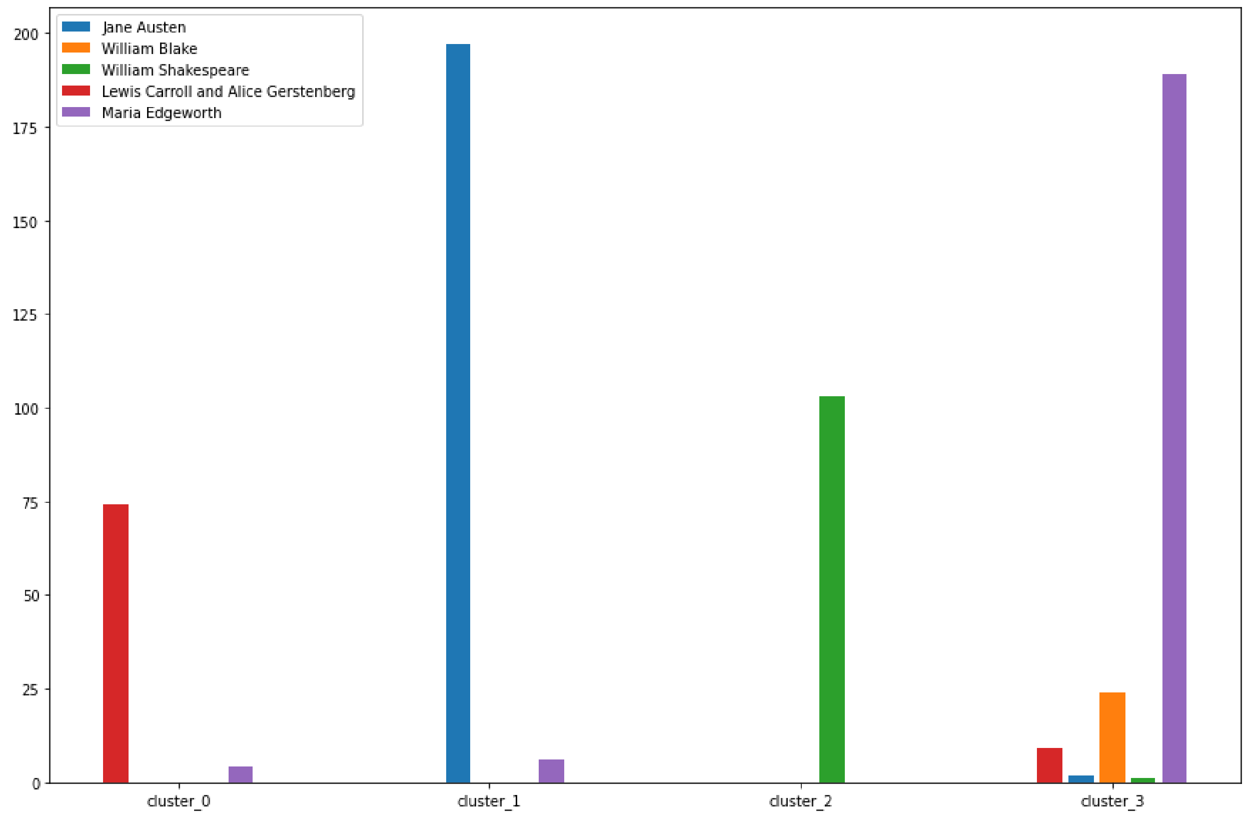


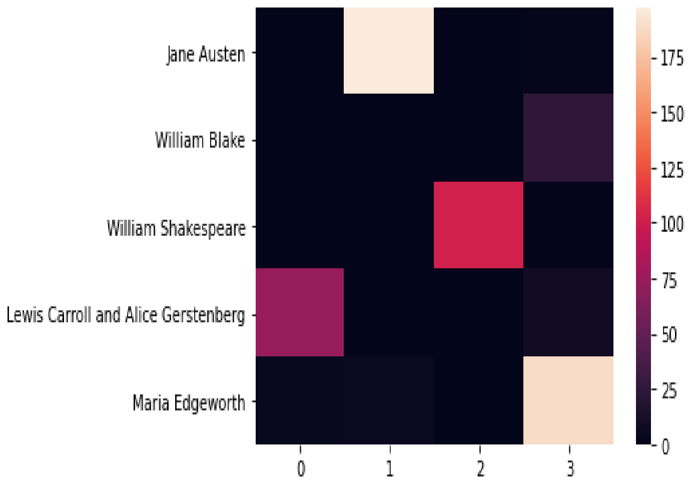


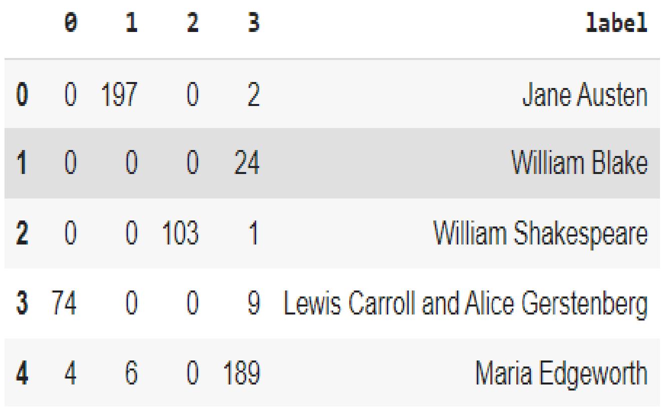
- Again we can see the pattern in the words “said” , “little” , “would” . These words are likely to appear in multiple stories and to mislead the algorithm.
- Words like “king” ,”Ham” , “Hamlet” are rare and they most likely belong to a single story or book.

Extra Visualizations

More visualizations of TF-IDF

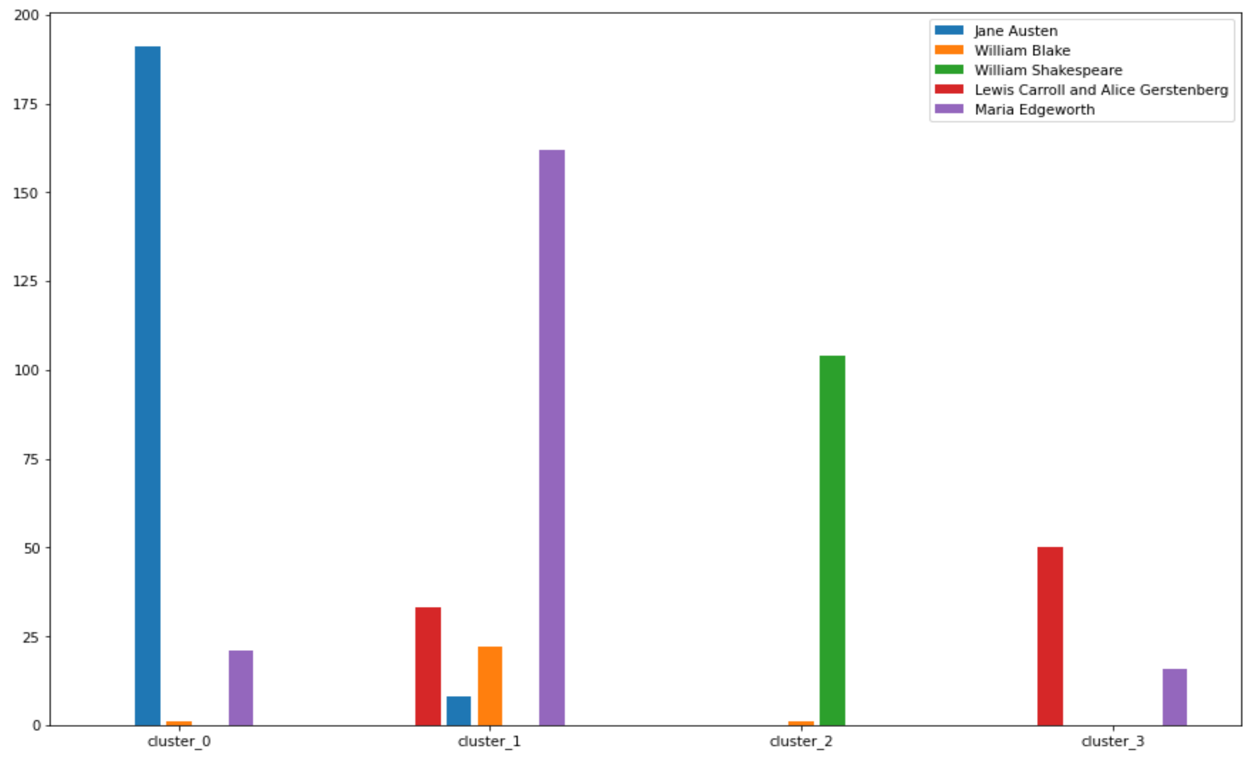


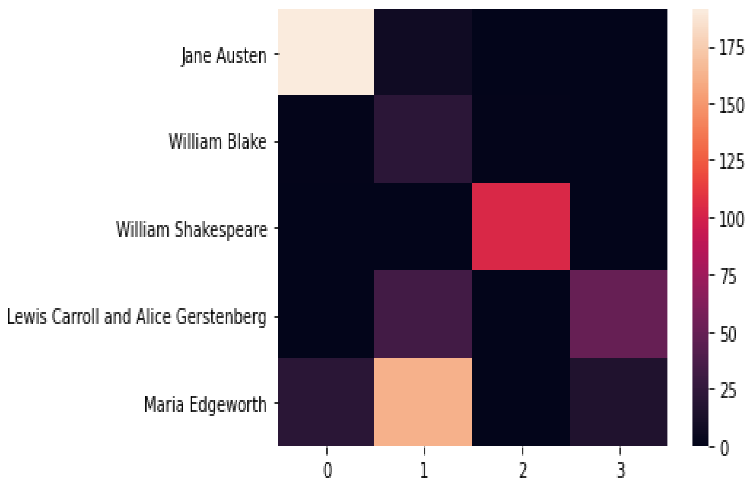


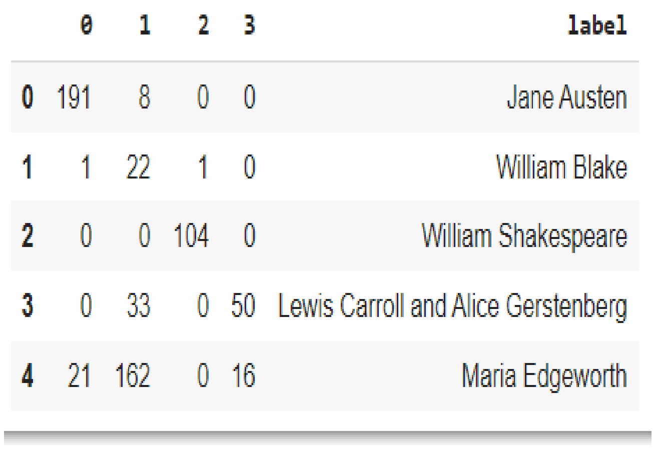


	0	1	2	3	label
0	0	197	0	2	Jane Austen
1	0	0	0	24	William Blake
2	0	0	103	1	William Shakespeare
3	74	0	0	9	Lewis Carroll and Alice Gerstenberg
4	4	6	0	189	Maria Edgeworth

More visualizations of BOW

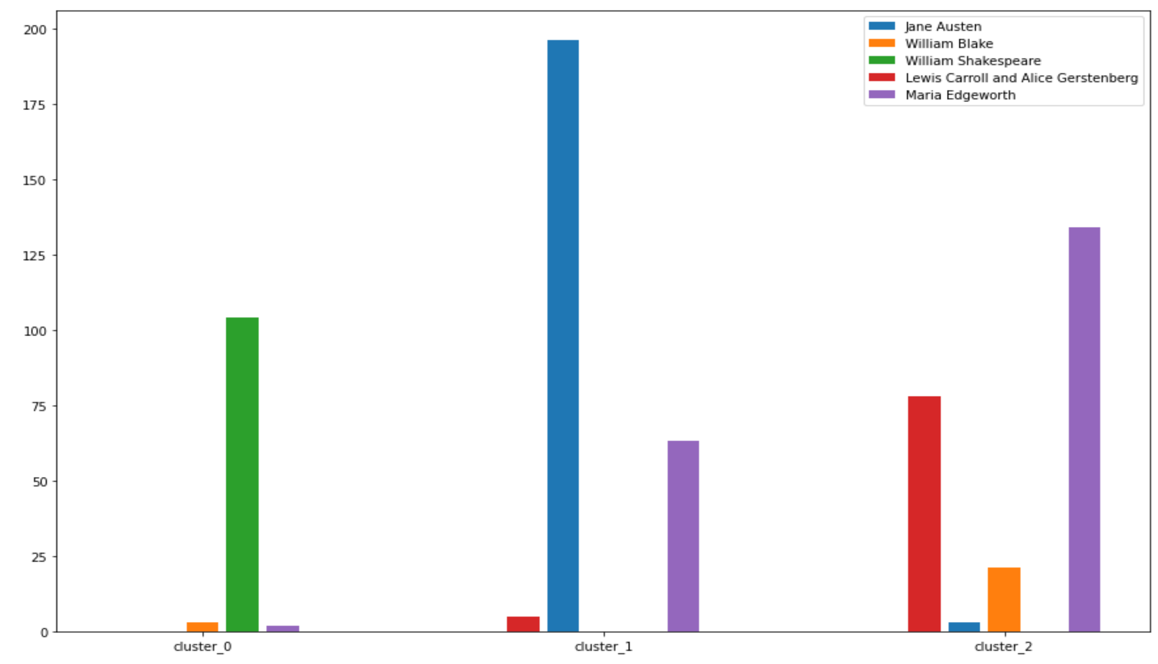


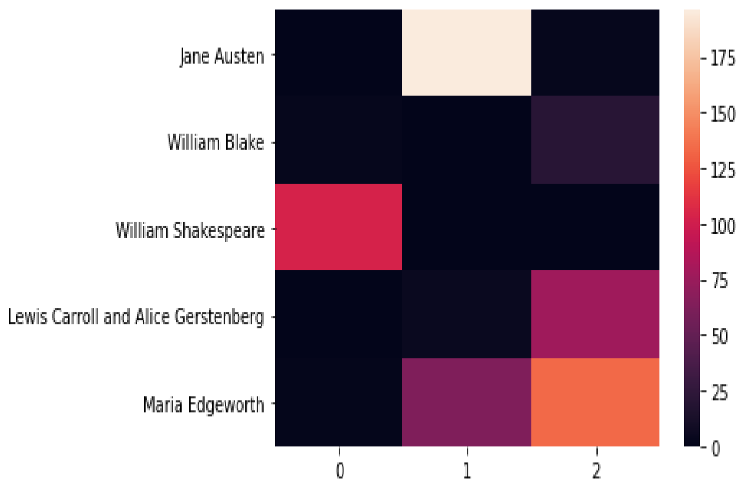


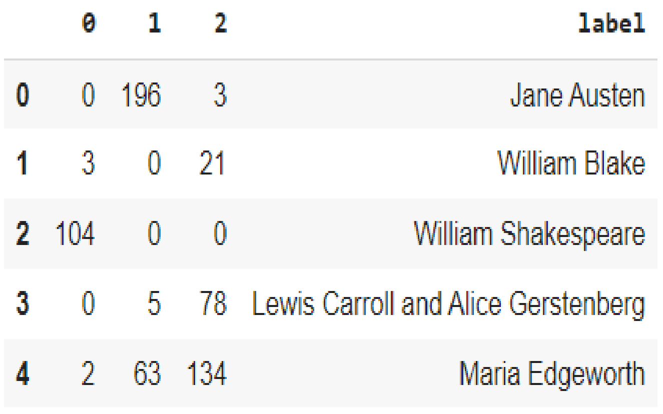


	0	1	2	3	label
0	191	8	0	0	Jane Austen
1	1	22	1	0	William Blake
2	0	0	104	0	William Shakespeare
3	0	33	0	50	Lewis Carroll and Alice Gerstenberg
4	21	162	0	16	Maria Edgeworth

More visualizations of Glove

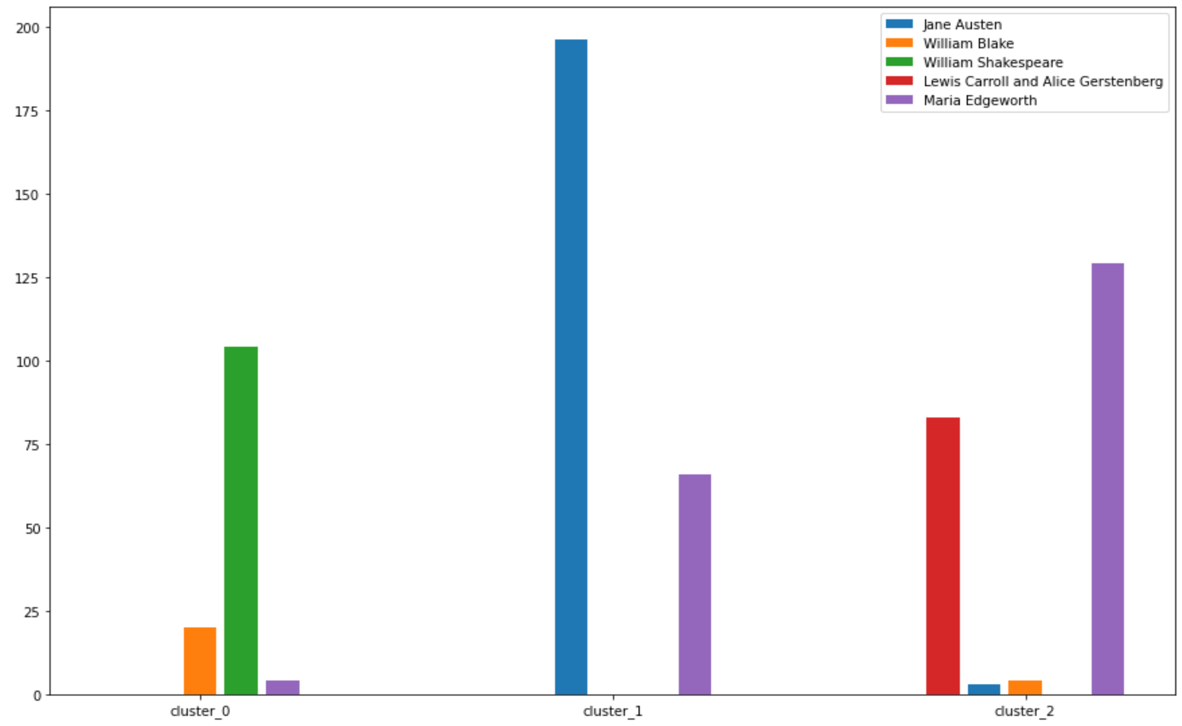


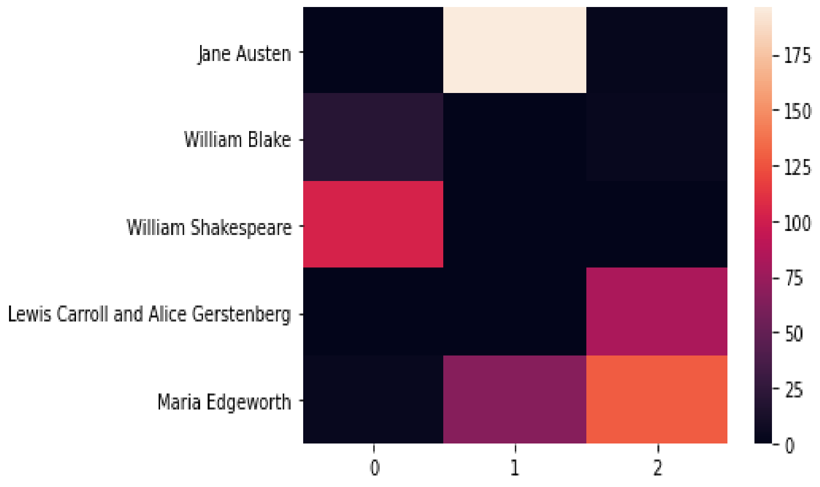


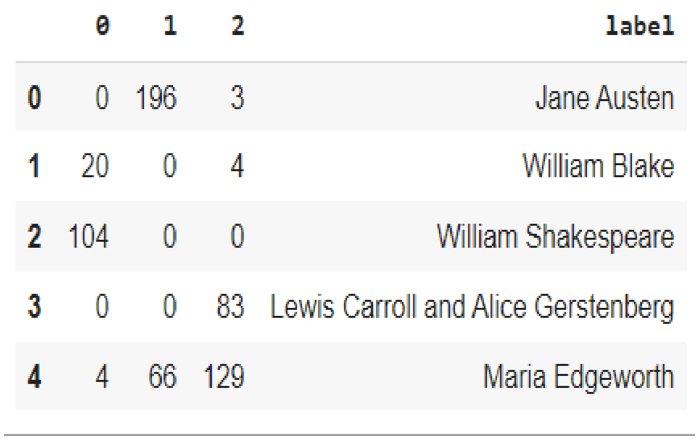


	0	1	2	label
0	0	196	3	Jane Austen
1	3	0	21	William Blake
2	104	0	0	William Shakespeare
3	0	5	78	Lewis Carroll and Alice Gerstenberg
4	2	63	134	Maria Edgeworth

More Visualizations of LDA







	0	1	2	label
0	0	196	3	Jane Austen
1	20	0	4	William Blake
2	104	0	0	William Shakespeare
3	0	0	83	Lewis Carroll and Alice Gerstenberg
4	4	66	129	Maria Edgeworth

Expectation–Maximization Algorithm (EM)

How the Algorithm Works

Expectation-Maximization algorithm can be used for the latent variables (variables that are not directly observable and are actually inferred from the values of the other observed variables) too in order to predict their values with the condition that the general form of probability distribution governing those latent variables is known to us.It is used to find the local maximum likelihood parameters of a statistical model in the cases where latent variables are involved and the data is missing or incomplete.

We used two feature extraction methods (TFiDF & BOW)

To determine the best n_components (Number of Clusters). There are two methods: Akaike Information Criteria (AIC) and Bayesian Information Criteria (BIC). The optimal number of clusters is the value that minimizes the AIC or BIC. We choose 4 clusters according to the graph below.

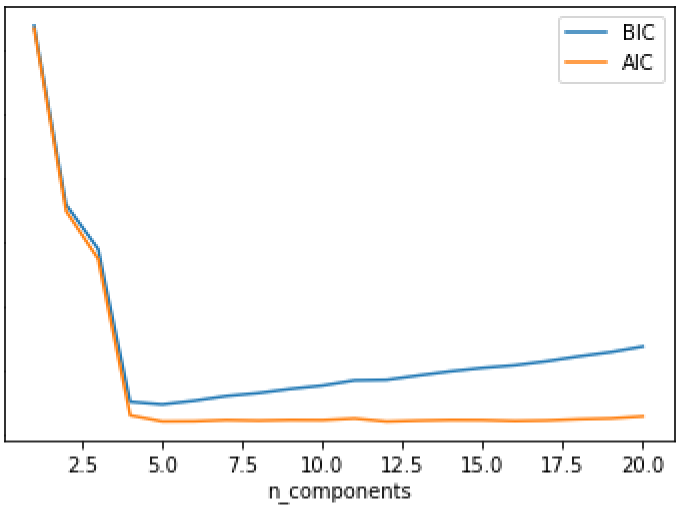


Figure for the distribution of clusters using Gaussian Mixture Model

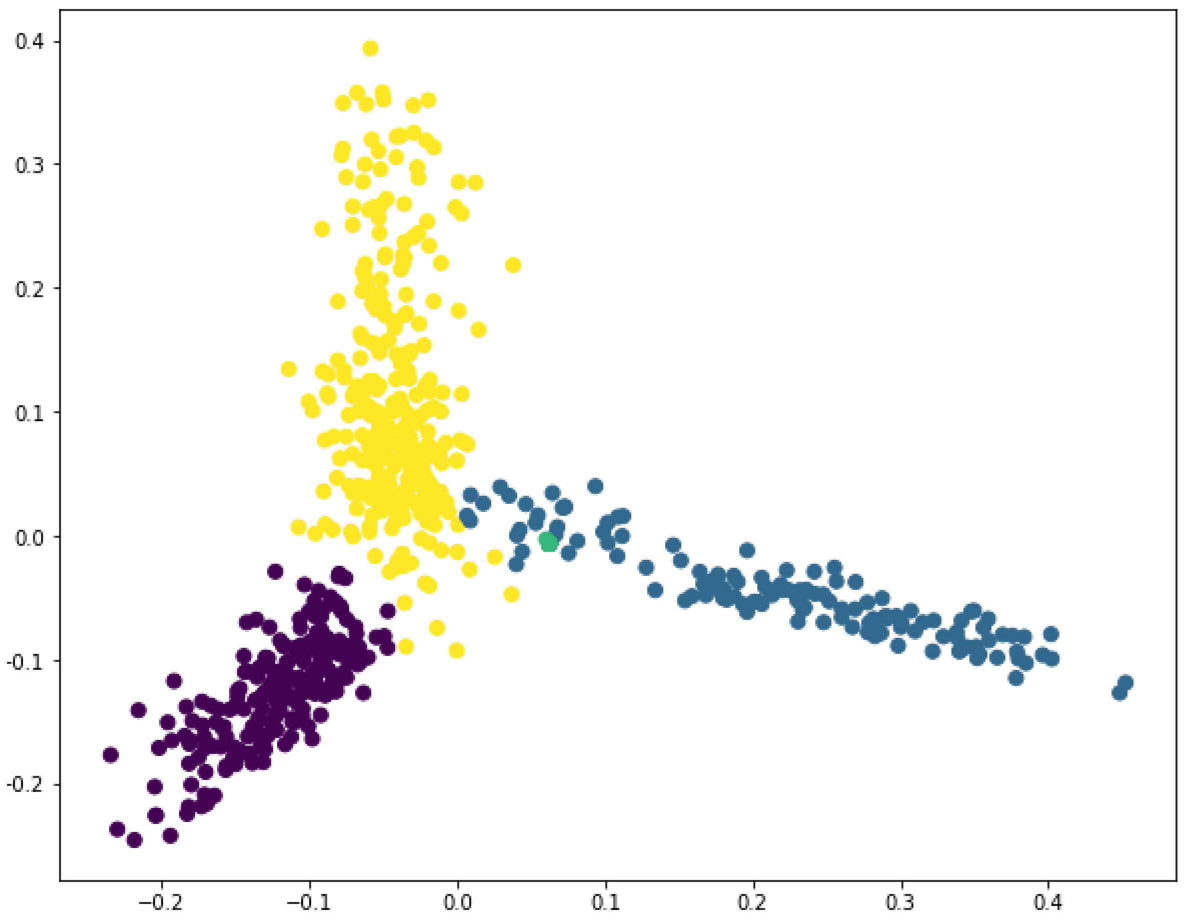
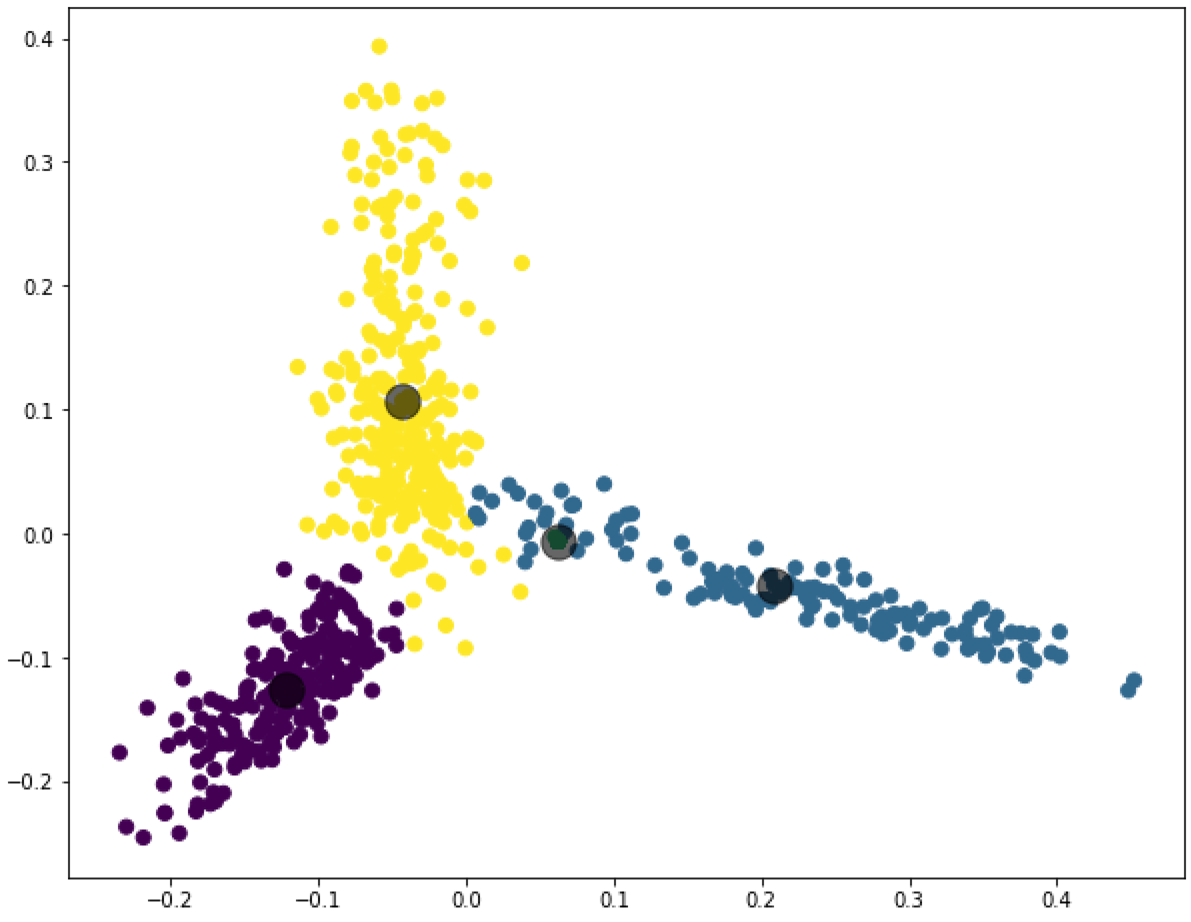


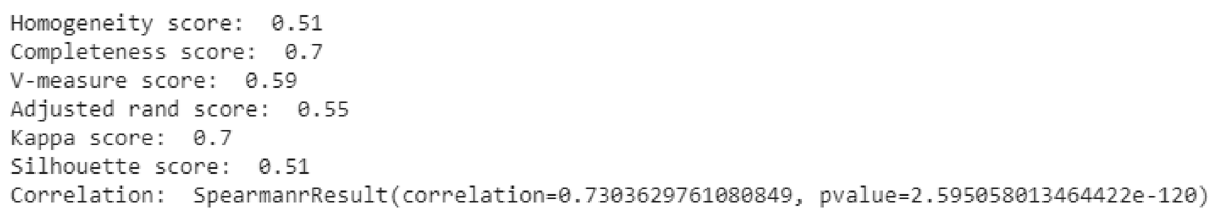
Figure to show the centroids of the clusters using the variables means and covariances to calculate the centers



TF-iDF Evaluation according to Kappa & Silhouette and other scores

```
Homogeneity score:  0.6
Completeness score:  0.82
V-measure score:  0.69
Adjusted rand score:  0.64
Kappa score:  0.76
Silhouette score:  0.54
Correlation:  SpearmanrResult(correlation=0.7728323432520907, pvalue=3.771397628276423e-143)
```

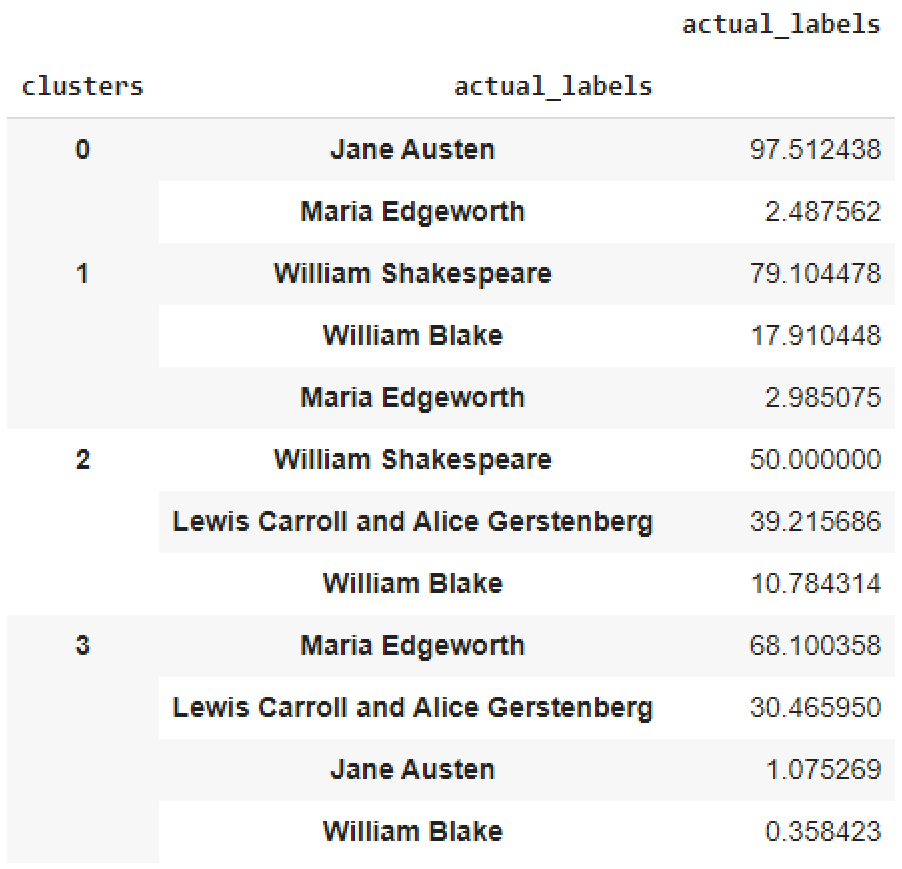
BOW Evaluation according to Kappa & Silhouette and other scores



```
Homogeneity score:  0.51
Completeness score:  0.7
V-measure score:  0.59
Adjusted rand score:  0.55
Kappa score:  0.7
Silhouette score:  0.51
Correlation:  SpearmanrResult(correlation=0.7303629761080849, pvalue=2.595058013464422e-120)
```

Human vs Predicted Labels in TF-iDF

In the below figure we can see that the author Jane Austen, the model can predict it well with a percentage of 97% after that William Shakespeare with a percentage 79%. These are the highest scores.



		actual_labels
clusters	actual_labels	
0	**Jane Austen**	97.512438
	Maria Edgeworth	2.487562
1	**William Shakespeare**	79.104478
	William Blake	17.910448
	Maria Edgeworth	2.985075
2	**William Shakespeare**	50.000000
	Lewis Carroll and Alice Gerstenberg	39.215686
	William Blake	10.784314
3	**Maria Edgeworth**	68.100358
	Lewis Carroll and Alice Gerstenberg	30.465950
	Jane Austen	1.075269
	William Blake	0.358423

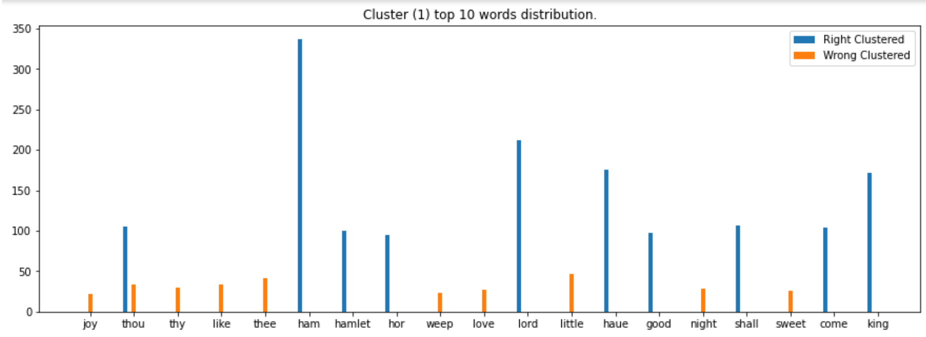
Perform Error Analysis

After assuming that the top author in each cluster represents the cluster label, we will consider this as the right clustered and the mis-clustered (mis-labeled) by other authors will be the wrong clusters.

So, We can conclude from the below graph to see the differences between those right vs wrong clusters examples to find out why the model got confused and considered them in one cluster and get out the common words between the right and wrong cluster.

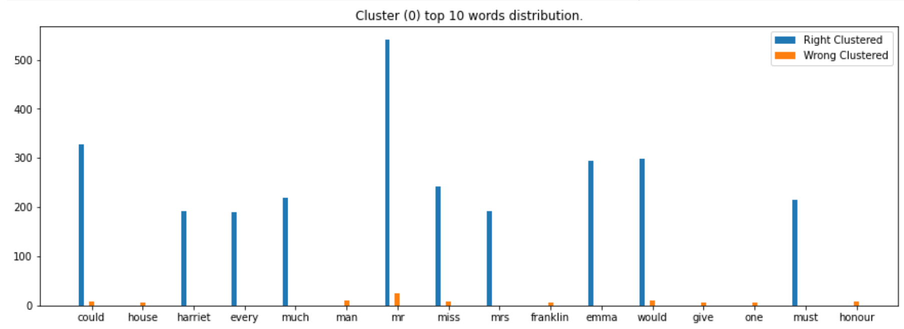
We will pick the highest confused clusters to show here in the report (you can find the whole visuals in the notebook)

Here's one common word 'thou' and this is the highest cluster 97%



```
Top 10 words in the right clustered
 [('ham', 337), ('lord', 212), ('haue', 175), ('king', 171), ('shall', 107), ('thou', 105), ('come', 104), ('hamlet', 100), ('good', 98), ('hor', 95)]

Top 10 words in the wrong clustered
 [('little', 47), ('thee', 41), ('thou', 34), ('like', 33), ('thy', 30), ('night', 28), ('love', 27), ('sweet', 26), ('weep', 23), ('joy', 22)]

Found 1 words in both right and wrong clustered top words [('thou', 34)]
```

Here's 4 common words in cluster(0) .



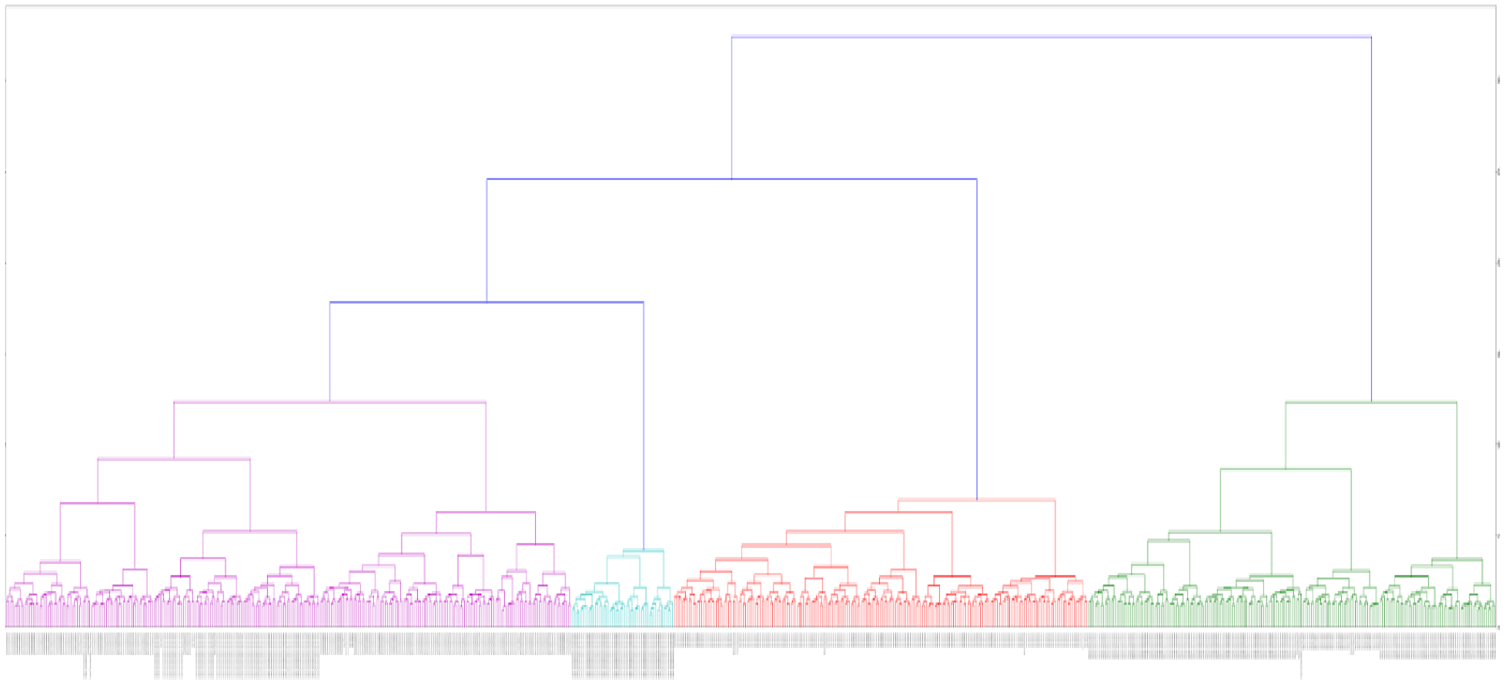
```
Top 10 words in the right clustered
 [('mr', 541), ('could', 327), ('would', 298), ('emma', 294), ('miss', 242), ('much', 218), ('must', 214), ('mrs', 192), ('harriet', 192), ('every', 189)]

Top 10 words in the wrong clustered
 [('mr', 24), ('would', 10), ('man', 10), ('could', 7), ('miss', 7), ('honour', 7), ('one', 6), ('franklin', 6), ('house', 5), ('give', 5)]

Found 4 words in both right and wrong clustered top words [('mr', 24), ('would', 10), ('could', 7), ('miss', 7)]
```

Hierarchical clustering Algorithms

How the Algorithm Works

Hierarchical clustering, also known as hierarchical cluster analysis, is an algorithm that groups similar objects into groups called clusters. The endpoint is a set of clusters, where each cluster is distinct from each other cluster, and the objects within each cluster are broadly similar to each other.

Hierarchical clustering typically works by sequentially merging similar clusters. This is known as agglomerative hierarchical clustering. In theory, it can also be done by initially grouping all the observations into one cluster, and then successively splitting these clusters. This is known as divisive hierarchical clustering. Divisive clustering is rarely done in practice.

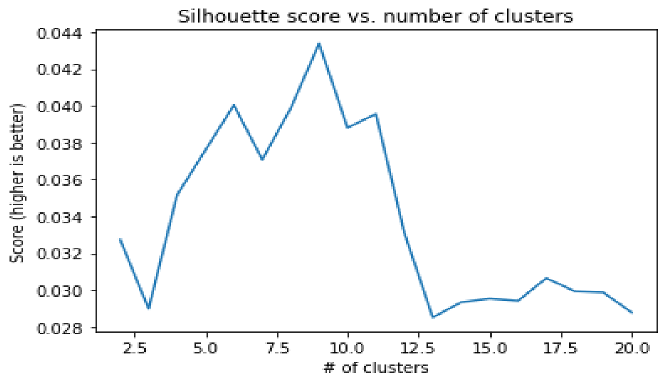


Choosing Number of Clusters

Choosing the number of clusters in a hierarchical algorithm is a little bit different , as we don't have centroids like Kmeans or EM algorithms that shape clusters around centroids and the distances between those vectors and the centroids.

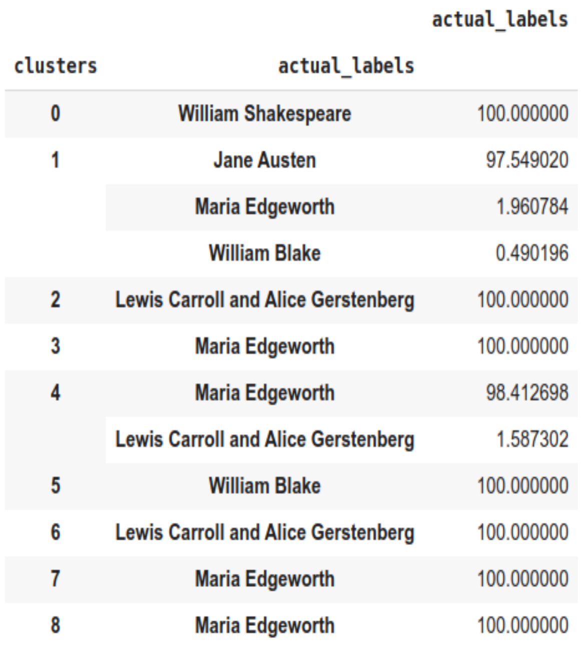
So, here we used a method that depends more on the silhouette score and picked the number of clusters with the highest silhouette score.

```
Silhouette score with 2 clusters: 0.03271856267578276
Silhouette score with 3 clusters: 0.02897998773797783
Silhouette score with 4 clusters: 0.03514699911085971
Silhouette score with 5 clusters: 0.037593489219300814
Silhouette score with 6 clusters: 0.04003345406482812
Silhouette score with 7 clusters: 0.03707948231946701
Silhouette score with 8 clusters: 0.03985200057709794
Silhouette score with 9 clusters: 0.04337894643763712
Silhouette score with 10 clusters: 0.03880737961107687
Silhouette score with 11 clusters: 0.03955048558514979
Silhouette score with 12 clusters: 0.033114278463355594
Silhouette score with 13 clusters: 0.028515909892244017
Silhouette score with 14 clusters: 0.029329729611001956
Silhouette score with 15 clusters: 0.029534811093420658
Silhouette score with 16 clusters: 0.029409545977138404
Silhouette score with 17 clusters: 0.03064140391744986
Silhouette score with 18 clusters: 0.02992837864777805
Silhouette score with 19 clusters: 0.029871440925702902
Silhouette score with 20 clusters: 0.02878092159033004
```

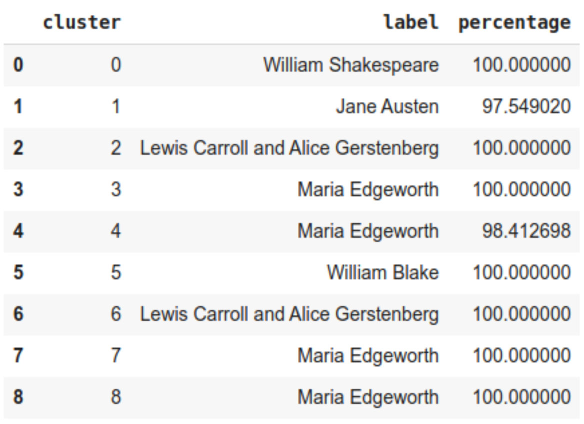


Human vs Predicted Labels

Comparing the clusters with the human labels and figuring out the distribution of each book/author among the 9 clusters.



		actual_labels
clusters	**actual_labels**	
0	**William Shakespeare**	100.000000
1	**Jane Austen**	97.549020
	Maria Edgeworth	1.960784
	William Blake	0.490196
2	**Lewis Carroll and Alice Gerstenberg**	100.000000
3	**Maria Edgeworth**	100.000000
4	**Maria Edgeworth**	98.412698
	Lewis Carroll and Alice Gerstenberg	1.587302
5	**William Blake**	100.000000
6	**Lewis Carroll and Alice Gerstenberg**	100.000000
7	**Maria Edgeworth**	100.000000
8	**Maria Edgeworth**	100.000000



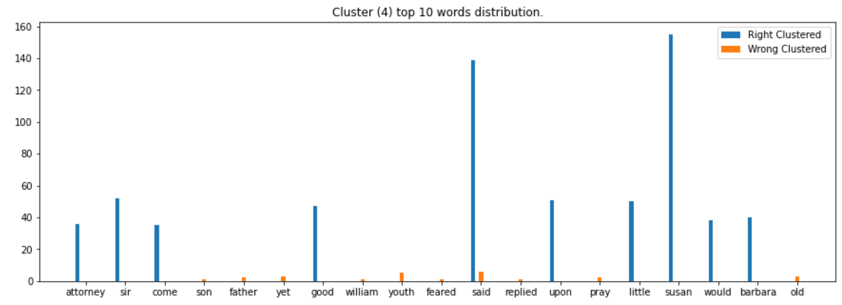
	cluster	label	percentage
0	0	William Shakespeare	100.000000
1	1	Jane Austen	97.549020
2	2	Lewis Carroll and Alice Gerstenberg	100.000000
3	3	Maria Edgeworth	100.000000
4	4	Maria Edgeworth	98.412698
5	5	William Blake	100.000000
6	6	Lewis Carroll and Alice Gerstenberg	100.000000
7	7	Maria Edgeworth	100.000000
8	8	Maria Edgeworth	100.000000

Perform Evaluations

```
Homogeneity score:  0.97
Completeness score:  0.97
V-measure score:  0.97
Adjusted rand score:  0.98
Kappa score:  0.99
Silhouette score:  0.02
Correlation:  SpearmanrResult(correlation=0.9889112060337032, pvalue=0.0)
```

Hierarchical algorithm achieves the best metrics, so far but it gives the lowest silhouette and that's the trade-off between metrics and you can't optimize the solution to have the highest values of both of them.

Perform Error Analysis

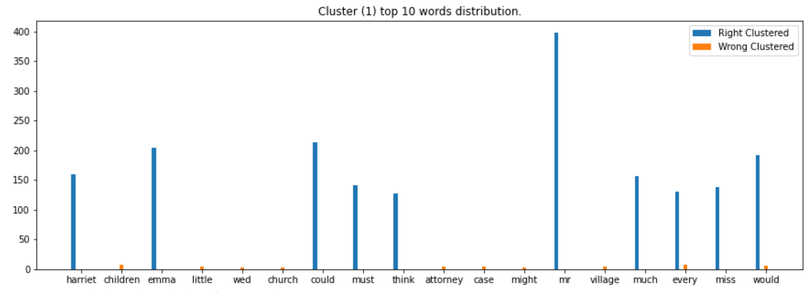


```
Top 10 words in the right clustered
 [('susan', 155), ('said', 139), ('sir', 52), ('upon', 51), ('little', 50), ('good', 47), ('barbara', 40), ('would', 38), ('attorney', 36), ('come', 35)]

Top 10 words in the wrong clustered
 [('said', 6), ('youth', 5), ('old', 3), ('yet', 3), ('pray', 2), ('father', 2), ('william', 1), ('replied', 1), ('son', 1), ('feared', 1)]

Found 1 words in both right and wrong clustered top words [('said', 6)]
```

As from the above figures we can conclude that books are mostly separated in different clusters and you can rarely find overlapped books/authors in the same cluster.

So, we can see that the top-frequent words are mostly ideally in the distribution.

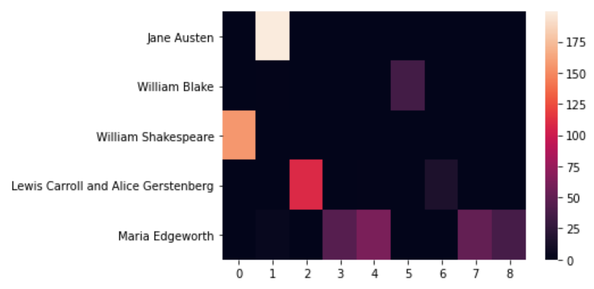
Another example cluster of the error analysis

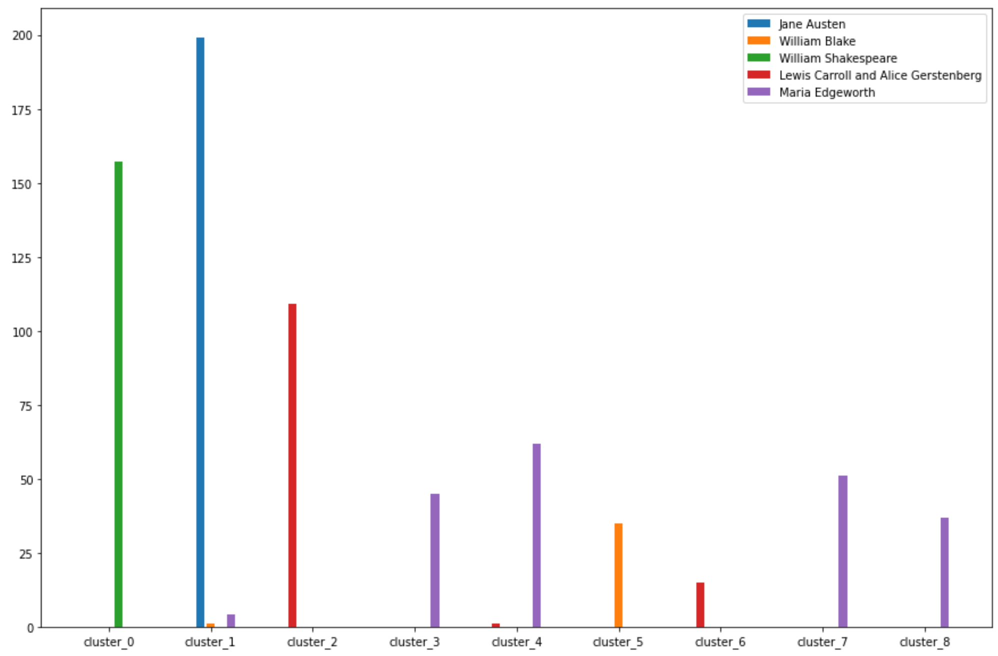


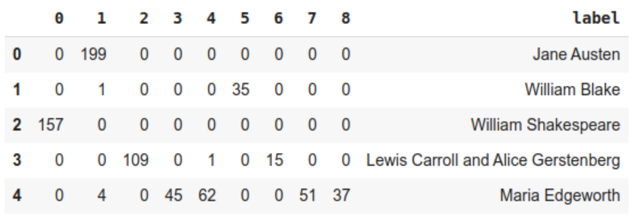
```
Top 10 words in the right clustered
 [('mr', 398), ('could', 213), ('emma', 204), ('would', 192), ('harriet', 160), ('much', 157), ('must', 141), ('miss', 138), ('every', 131), ('think', 127)]

Top 10 words in the wrong clustered
 [('children', 7), ('every', 7), ('would', 6), ('little', 5), ('village', 5), ('attorney', 4), ('case', 4), ('wed', 3), ('church', 3), ('might', 3)]

Found 2 words in both right and wrong clustered top words [('every', 7), ('would', 6)]
```

This is the most one with the overlapped words between the right and wrong clustered.

Further Visualizations







	0	1	2	3	4	5	6	7	8	label
0	0	199	0	0	0	0	0	0	0	Jane Austen
1	0	1	0	0	0	35	0	0	0	William Blake
2	157	0	0	0	0	0	0	0	0	William Shakespeare
3	0	0	109	0	1	0	15	0	0	Lewis Carroll and Alice Gerstenberg
4	0	4	0	45	62	0	0	51	37	Maria Edgeworth

This distribution graph shows how the authors/books are distributed among the 9 clusters. We can see that **Shakespeare** and **Jane Austen** books are well separated from any other clusters. **Maria Edgeworth** is mostly distributed among 4 clusters causing a little bit of distortion of its cluster assignment.

LDA Algorithms

How the Algorithm Works

The Latent Dirichlet Allocation (LDA) algorithm is used mainly for topic modelling to find out the best topics and the best predicted labels for each cluster you can run for.

- Here we used LDA as our last approach to see how we can model our 5 books in 5 clusters.
- We didn't use a method for number of topics/clusters detection like elbow method, we just need to model our data to 5 topics and we need to see how the LDA could figure out if the 5 topics could match the 5 books or not.

LDA model

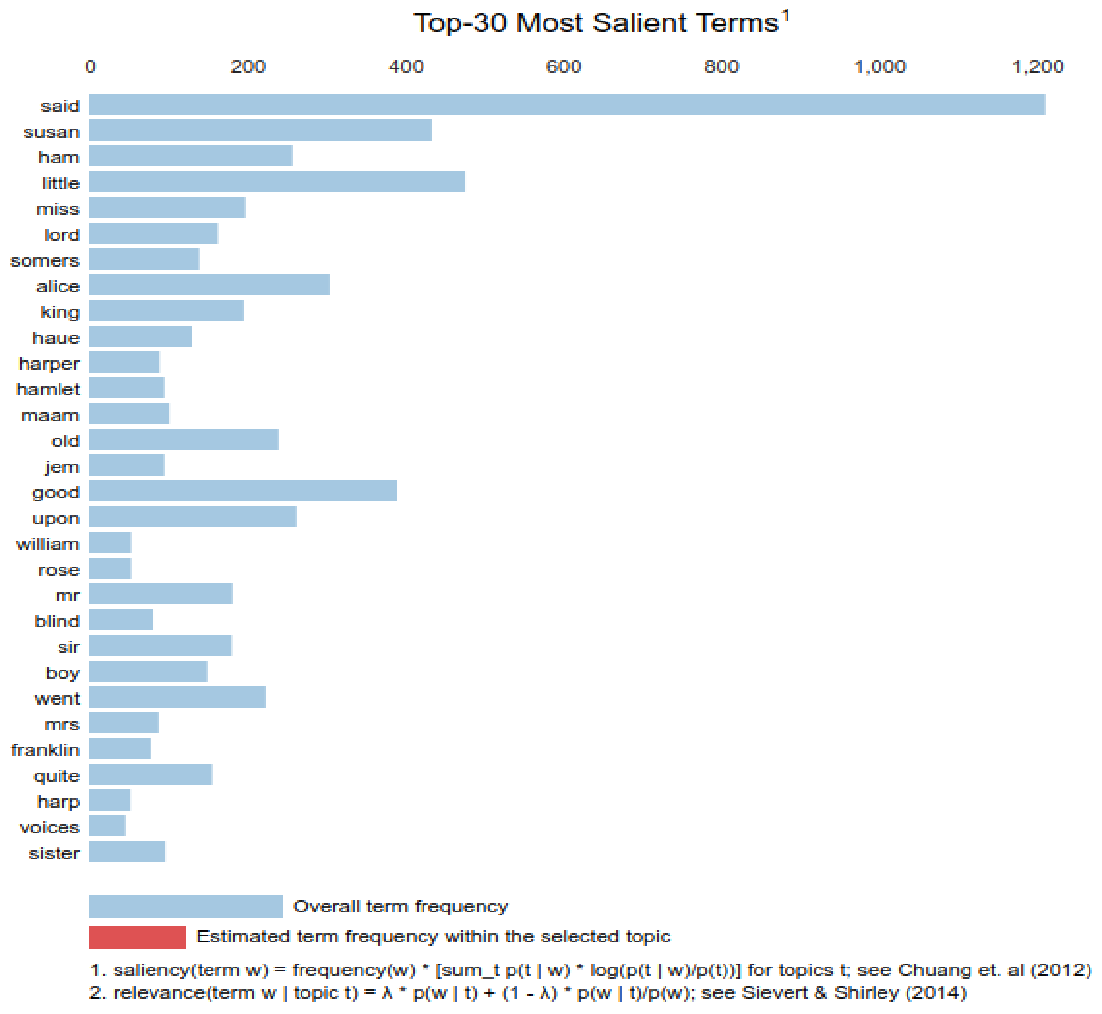
```
# Create a corpus from a list of texts
common_dictionary = Dictionary(lines_words)
common_corpus = [common_dictionary.doc2bow(text) for text in lines_words]

# Train the model on the corpus.
lda_model = LdaModel(corpus=corpus,
                     id2word=id2word,
                     num_topics=5,
                     random_state=100,
                     update_every=1,
                     chunksize=100,
                     passes=10,
                     alpha='auto',
                     per_word_topics=True)
```

Topics top-words distribution

```
[(0,
  '0.028*"susan" + 0.011*"harper" + 0.008*"rose" + 0.008*"william" + 0.008*"little" + 0.007*"voices" + 0.007*"blind" + 0.006*"thy" + 0.006*"old" + 0.006*"lamb"'),
 (1,
  '0.015*"susan" + 0.011*"good" + 0.010*"miss" + 0.008*"upon" + 0.008*"somers" + 0.007*"sir" + 0.007*"would" + 0.007*"mr" + 0.007*"us" + 0.006*"maam"'),
 (2,
  '0.037*"said" + 0.013*"little" + 0.009*"alice" + 0.009*"one" + 0.009*"know" + 0.008*"could" + 0.007*"would" + 0.007*"went" + 0.007*"see" + 0.006*"much"'),
 (3,
  '0.009*"miss" + 0.008*"susan" + 0.007*"somers" + 0.007*"harp" + 0.006*"answered" + 0.006*"standing" + 0.005*"simple" + 0.004*"grew" + 0.004*"blush" + 0.004*"bumper"'),
 (4,
  '0.018*"ham" + 0.011*"lord" + 0.010*"king" + 0.009*"haue" + 0.007*"hamlet" + 0.006*"good" + 0.006*"come" + 0.005*"shall" + 0.005*"let" + 0.005*"thou"')]
```

- Here we can see that topic (4) is so distinctive in its language as it uses mainly old english
- The word "Susan" is repeated in the top two words in three clusters (0, 1, 3)

Top-30-salient terms



The intent of salience is to help identify which words are the most informative words for identifying topics in all the documents. Higher saliency values indicate that a word is more useful for identifying a specific topic. Saliency is always a positive value, and it does not have a maximum. A value of 0 indicates that a given word effectively belongs equally to all topics. Saliency is designed to look at words on a corpus-scale, as opposed to individual topic levels.

Performance Evaluations

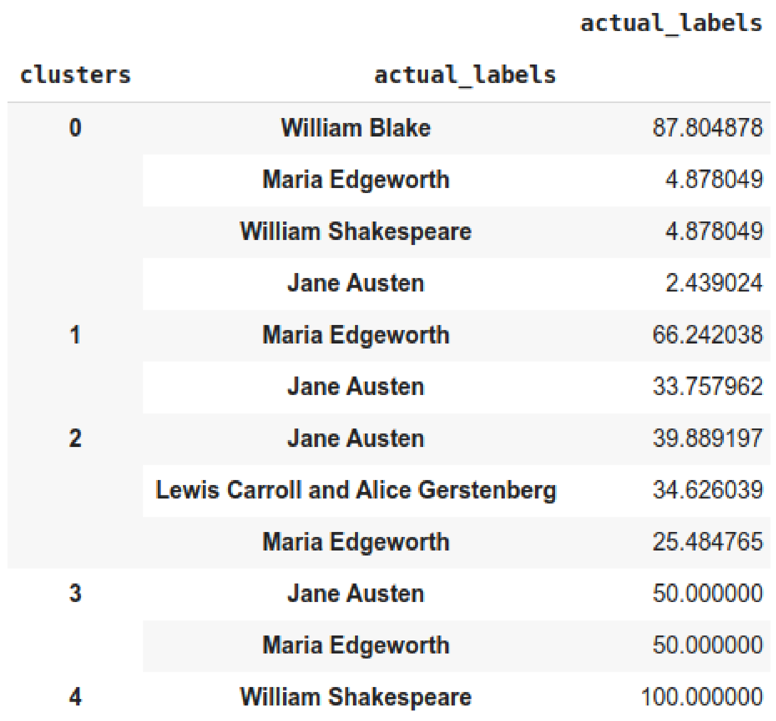
```
Homogeneity score:  0.52
Completeness score:  0.67
V-measure score:  0.59
Adjusted rand score:  0.38
Kappa score:  0.66
Silhouette score:  0.01
Correlation:  SpearmanrResult(correlation=0.7508590971944392, pvalue=9.342580150251219e-131)
```

- It has a **high** Kappa-score but **low** Silhouette score , and that’s the trade-off between the two metrics that could happen for many models.
- We can also see relevant homogeneous and strong correlation with the cluster labels and the actual labels (author labels)

Human Labels Vs Predicted Labels

Clusters distribution to author labels

A distribution of the author in each book

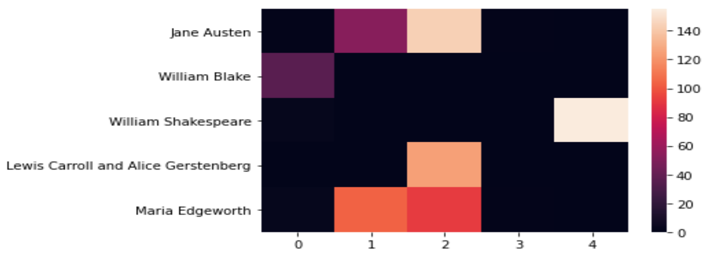


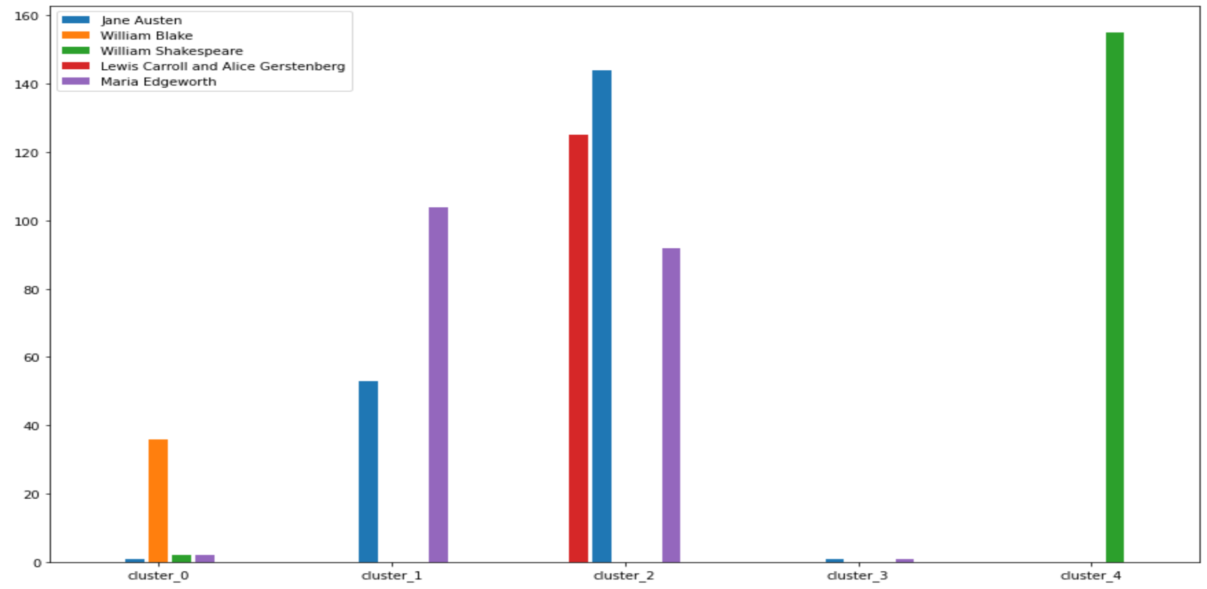
		actual_labels
clusters	**actual_labels**	
0	**William Blake**	87.804878
	Maria Edgeworth	4.878049
	William Shakespeare	4.878049
	Jane Austen	2.439024
1	**Maria Edgeworth**	66.242038
	Jane Austen	33.757962
2	**Jane Austen**	39.889197
	Lewis Carroll and Alice Gerstenberg	34.626039
	Maria Edgeworth	25.484765
3	**Jane Austen**	50.000000
	Maria Edgeworth	50.000000
4	**William Shakespeare**	100.000000

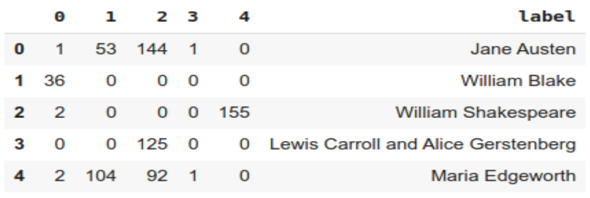
- It is clear that the “William Shakespeare” book is very distinctive and can be visualized in a separate cluster.
- We use here the top book for each cluster as its label for this cluster for example for cluster_0 we considered it labelled as for “William Blake” author.
- We can see that the cluster can’t catch all the percentages
- The perfect cluster is to have 100% percentage for each separate book for it.

Labels distribution to the clusters

Here we can see three different representations of how the labels are distributed to the clusters.







	0	1	2	3	4	label
0	1	53	144	1	0	Jane Austen
1	36	0	0	0	0	William Blake
2	2	0	0	0	155	William Shakespeare
3	0	0	125	0	0	Lewis Carroll and Alice Gerstenberg
4	2	104	92	1	0	Maria Edgeworth

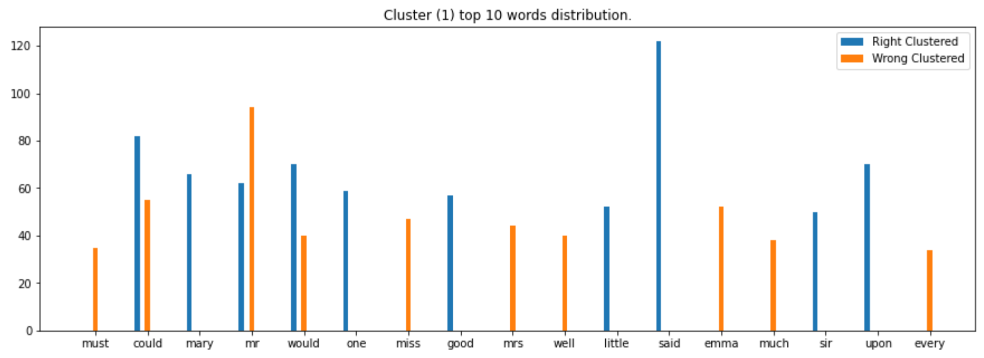
- “Maria Edgeworth” has the highest distribution among all authors; it has the highest distribution of 4 clusters out of 5 and 2 of them have large distributions.
- “Shakespeare” also can be seen here as it is the most distinctive book.

Perform Error Analysis

After assuming that the top author in each cluster represents the cluster label, we will consider this as the right clustered and the mis-clustered (mis-labeled) by other authors will be the wrong clustered.

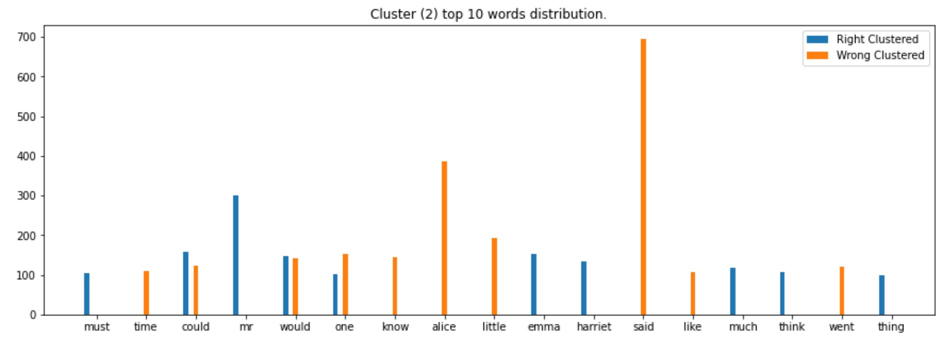
So, We can conclude from the below graph to see the differences between those right vs wrong clustered examples to find out why the model got confused and considered them in one cluster.

We will pick the highest confused clusters to show here in the report (you can find the whole visuals in the notebook)



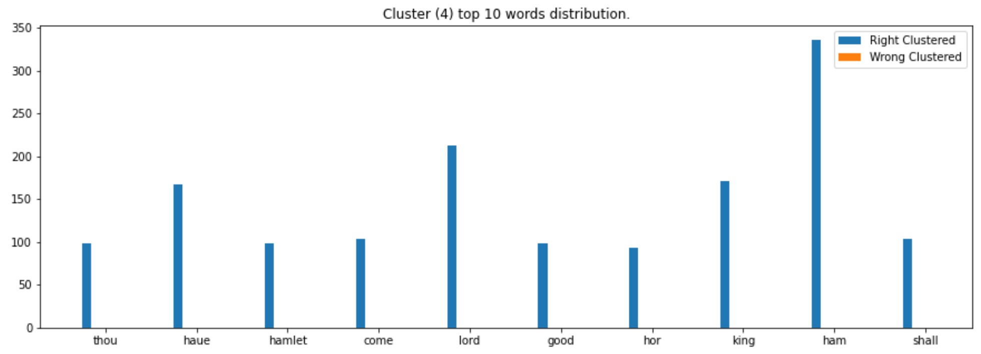
```
Top 10 words in the right clustered
 [('said', 122), ('could', 82), ('would', 70), ('upon', 70), ('mary', 66), ('mr', 62), ('one', 59), ('good', 57), ('little', 52), ('sir', 50)]

Top 10 words in the wrong clustered
 [('mr', 94), ('could', 55), ('emma', 52), ('miss', 47), ('mrs', 44), ('would', 40), ('well', 40), ('much', 38), ('must', 35), ('every', 34)]

Found 3 words in both right and wrong clustered top words [('mr', 94), ('could', 55), ('would', 40)]
```

- We can see here the confusions happened in clusters (1) that it intersects in top 3 words as well as some of those words are salient terms that can represent the topics so they have high weight to assign documents to topics.



```
Top 10 words in the right clustered
 [('mr', 301), ('could', 157), ('emma', 152), ('would', 148), ('harriet', 134), ('much', 118), ('think', 106), ('must', 104), ('one', 102), ('thing', 99)]

Top 10 words in the wrong clustered
 [('said', 695), ('alice', 385), ('little', 193), ('one', 153), ('know', 144), ('would', 142), ('could', 124), ('went', 119), ('time', 109), ('like', 108)]

Found 3 words in both right and wrong clustered top words [('one', 153), ('would', 142), ('could', 124)]
```

- Same as above example in cluster(4) , here in cluster (2) we can see also that it intersect in three words and made the cluster confused a little bit with the right clustered authors, but the main the difference here we can see that those terms has low frequency, so this clusters has less distributions of wrong clusters than in cluster (4)

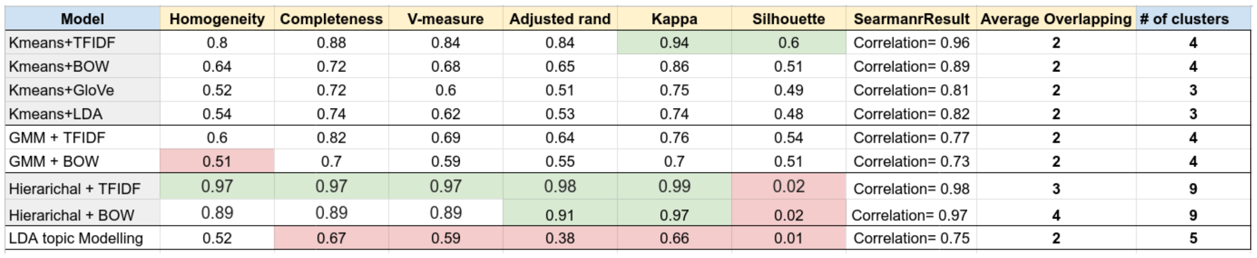


```
Top 10 words in the right clustered
 [('ham', 336), ('lord', 212), ('king', 171), ('haue', 167), ('come', 104), ('shall', 104), ('thou', 99), ('hamlet', 98), ('good', 98), ('hor', 93)]

Top 10 words in the wrong clustered
 []

Found 0 words in both right and wrong clustered top words []
```

- This is an example of the optimal case , that you don't have any wrong-clustered examples in the cluster and you successfully clustered the label(author) perfectly.

Comparing Different Models

- **Average Overlapping:** It is our metric to understand more the books distribution among clusters, the optimal value is (0) this means that each cluster perfectly separated each book in its own cluster, and the worst value is (5) which is the number of books this means that all the books are overlapped on all clusters.



Model	Homogeneity	Completeness	V-measure	Adjusted rand	Kappa	Silhouette	SearmanrResult	Average Overlapping	# of clusters
Kmeans+TFIDF	0.8	0.88	0.84	0.84	0.94	0.6	Correlation= 0.96	**2**	**4**
Kmeans+BOW	0.64	0.72	0.68	0.65	0.86	0.51	Correlation= 0.89	**2**	**4**
Kmeans+GloVe	0.52	0.72	0.6	0.51	0.75	0.49	Correlation= 0.81	**2**	**3**
Kmeans+LDA	0.54	0.74	0.62	0.53	0.74	0.48	Correlation= 0.82	**2**	**3**
GMM + TFIDF	0.6	0.82	0.69	0.64	0.76	0.54	Correlation= 0.77	**2**	**4**
GMM + BOW	0.51	0.7	0.59	0.55	0.7	0.51	Correlation= 0.73	**2**	**4**
Hierarichal + TFIDF	0.97	0.97	0.97	0.98	0.99	0.02	Correlation= 0.98	**3**	**9**
Hierarichal + BOW	0.89	0.89	0.89	0.91	0.97	0.02	Correlation= 0.97	**4**	**9**
LDA topic Modelling	0.52	0.67	0.59	0.38	0.66	0.01	Correlation= 0.75	**2**	**5**

We can say that our champion model could be the KMeans with TF-IDF

As it achieves the balance between kappa and silhouette and also for other scores, besides that it has also the lowest average overlapping of books between clusters

Conclusion

- **Maria Edgeworth's book** (Children's stories) is the **most one that overlapped between all clusters**, it intersects with other books in many meanings and words
- **Shakespeare's book** is the **most distinctive book among all books**, it always tends to be separated in its own cluster usually, because the use of old english language and the vocab which is not common nowadays.
- There is a **trade off** between **the number of clusters** and the **silhouette score**, the **average overlapping** is also associated with the number of clusters as it increases the overlapping also increases.
- **[Future work]** Enhance the features engineering more
- **[Future work]** Develop an ensemble model for clustering to gather different clustering algorithms in one predictive model
- **[Future work]** Extend the partitions and take more descriptive sentences from the books