Amazon Fine Foods Reviews: Text Clustering

Madalina-Mihaela Badescu

January 25, 2023

1 Introduction

In this paper I will present my approach and my results when applying two unsupervised models on a dataset containing reviews. The dataset that I chose is made of reviews for fine foods from Amazon, together with information about the reviewer, the product and the score that each user gave to each product. For the training of the unsupervised learning algorithms I used the texts' of each review.

The models that I chose to work with are K-Means clustering and Agglomerative clustering. My approach for finding the best results for the unsupervised learning algorithms was to use the Elbow Method, Grid Search and the Silhouette Coefficient. The models were trained on two types of features: Bag of Words (BoW) and term frequency–inverse document frequency (TF-IDF).

After compiling each algorithm, I made plots in order to better visualize the data and the cluster distribution. I compared all this information and the overall results with random choice and also with a supervised model (SVM). I concluded that both algorithms performed better than the random choice, but worse than the SVM model.

2 Processing the data

For the processing of the data, I used a number of Natural Language Processing (NLP) methods, in order to compute an improved set of texts. The steps that I used were:

- I started by removing the duplicates. This was done because of the redundancy of data that appears twice. I started with a set of 10000 reviews and, after the removing of the duplicates, I had left 9515 rows.
- Secondly, I analyzed the distribution of the reviews based on score and, for that, I computed Figure 1.
- The next step was removing html tags. These special tags are not relevant to the score or message of the review, so they are not necessary.
- Then, I analyzed the text that was left and proceeded to eliminate all punctuation signs. Commas, points and so on are not relevant for a machine learning algorithm, so it is better to remove them.

- Afterwards, for further processing, I needed all the texts to be separated into words, so I could continue the augmentation of the data. Thus, I made all words of each review into tokens.
- Next, also in preparation for the further steps, I modified all letters to be lower case.
- Using the preparations that I've done, I removed all stop words from the dataset with the function nltk.corpus.stopwords. The words that were removed due to this action can be seen in Figure 2. Stop words are deleted because they don't convey any important message to the algorithm and, instead, take up processing time.
- Lastly, for the prepocessing of data, I used the lemmatization method. This procedure takes each word from the form that it is in each text and transforms it into its dictionary form. For this, I used the SpaCy lemmatization, as it analyzes which part of speech each word is before transforming it.

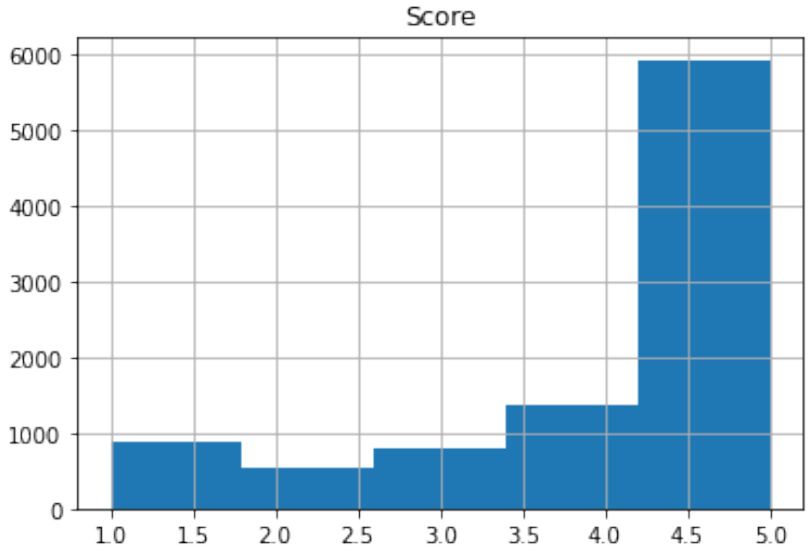

Figure 1: Score Histogram

```
['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours',
'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'its
elf', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these',
'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doin
g', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'o
ut', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'al
l', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so',
'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o',
're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'has
n', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'sha
n', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]
```

Figure 2: Stop words set

In the end, I joined all the processed tokens into separate reviews.

3 Feature representation

As part of feature representation, I chose to use to methods: BoW and TF-IDF. Afterwards, I trained each model on each separate type of feature, in order to analyze which one performed better.

Bag of Words is a form of feature representation that remembers the number of times that each word is used in a text. This is primarily used for classification or clustering, but it can be useful in a large number of NLP problems. In my project, I used BoW only for separate words, but it can be used also as a n-gram, meaning it can compute the multiplicity of pairs of words or bigger groups.

Term frequency–inverse document frequency is a feature representation that computes the weight that each word has, relating it to its frequency. It is used in a lagre number of search engines and it's useful in most algorithm regarding text classification or clustering. I used it as the second feature extraction technique for my algorithms.

4 K-Means model

4.1 Definition of the model

K-Means is a clustering model that uses as metrics the distance between points in order to compute the results. The algorithm receives a number of clusters that it must split the data into and minimizes the inertia. K-Means performs better or worse based on the number of clusters that it's programmed to use, so usually it's good to perform parameter fine-tuning, in order to find an optimal k.

4.2 Elbow Method

The Elbow Method is used in unsupervised learning, in order to find the best number of clusters for parting the dataset. The cluster number is found where the curve of the graph has its "elbow". In practice, it is not always very clear what is the perfect choice, so it is good to use more than one method in order to determine the best value for k.

In Figure 3 and Figure 4 we can see that the computation of KElbowVisualizer for YellowBrick determined that the best k for BoW is 5 and for TF-IDF is 6, but the curves don't have a very visible elbow. The figures represent the results on both types of features.

If we want to interpret Figure 3 and Figure 4, we can say that there is not a very clear way of splitting the number of clusters, but there can be a change with $k = 5$ or $k = 6$.

4.3 Grid Search

To further investigate our best parameters and fine-tune, we computed a grid search. This method takes a set of parameters for which to try different values in all possible combinations. In the end, it computes the best choices and the best score.

In Figure 5 there are some highlights of my grid search for the BoW feature. We can observe that the best set of features is {'maxiter': 300, 'nclusters': 6, 'ninit': 15, 'tol': 0.0001} and I computed that the best score is -105805.18. And in Figure 6 there

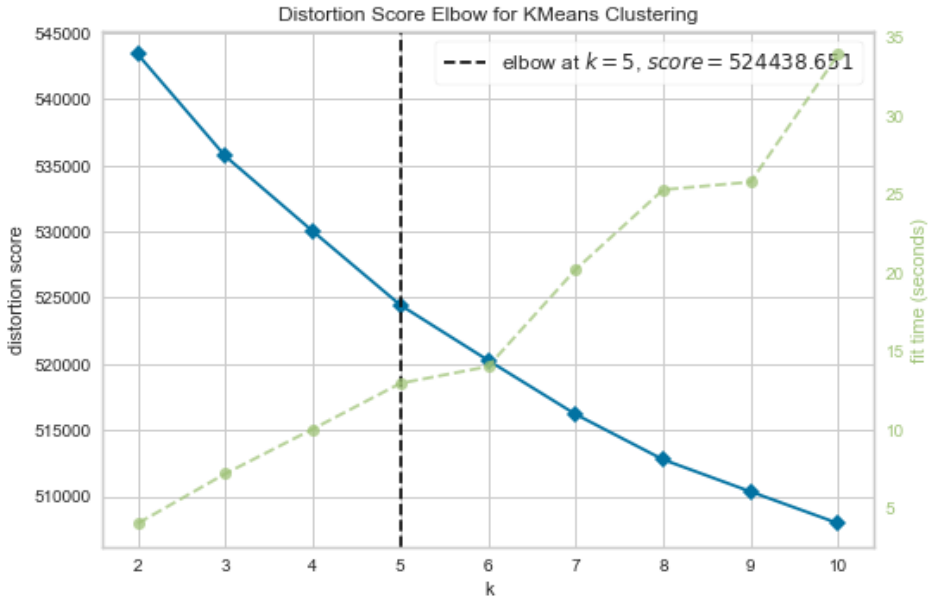

Figure 3: Stop words set

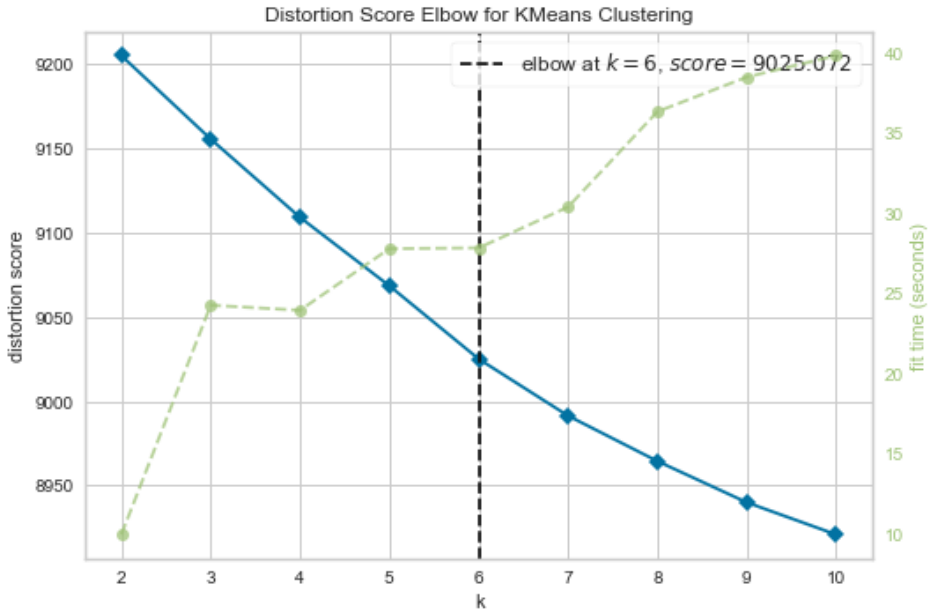

Figure 4: Stop words set

are some highlights of my grid search for the TF-IDF feature. We can observe that the best set of features is {'maxiter': 300, 'nclusters': 6, 'ninit': 1, 'tol': 0.0001} and I computed that the best score is -1827.06.

```
[CV 2/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-102511.037 total time=   7.6s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-107829.102 total time=   5.2s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-113569.565 total time=   5.7s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-105376.055 total time=   8.9s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-06;, score=-102567.156 total time=  11.8s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-06;, score=-102507.968 total time=  12.6s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-06;, score=-105759.811 total time=  12.9s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-06;, score=-113684.009 total time=  11.4s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-06;, score=-106350.685 total time=  14.2s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-101565.861 total time=  15.8s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-103754.556 total time=  12.3s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-106229.886 total time=  14.0s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-113210.726 total time=  13.2s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-106323.554 total time=  15.2s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-102382.349 total time=  17.4s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-102516.068 total time=  15.9s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-105845.996 total time=  13.4s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-113926.257 total time=  11.4s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-106300.594 total time=  15.2s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-06;, score=-101576.096 total time=  16.5s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-06;, score=-103142.407 total time=  20.0s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-06;, score=-105945.503 total time=  16.5s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-06;, score=-113301.888 total time=  19.0s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-06;, score=-105374.663 total time=  23.2s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-05;, score=-101572.325 total time=  20.2s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-05;, score=-102687.831 total time=  21.0s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-05;, score=-106235.771 total time=  18.7s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-05;, score=-113250.389 total time=  19.5s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=15, tol=1e-05;, score=-105390.936 total time=  19.8s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=15, tol=0.0001;, score=-101599.290 total time=  19.9s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=15, tol=0.0001;, score=-102516.895 total time=  21.5s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=15, tol=0.0001;, score=-105953.437 total time=  19.1s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=15, tol=0.0001;, score=-113400.669 total time=  18.2s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=15, tol=0.0001;, score=-105555.638 total time=  17.3s
[CV 1/5] END max_iter=400, n_clusters=4, n_init=1, tol=1e-06;, score=-103076.848 total time=   0.8s
[CV 2/5] END max_iter=400, n_clusters=4, n_init=1, tol=1e-06;, score=-104091.828 total time=   0.6s
[CV 3/5] END max_iter=400, n_clusters=4, n_init=1, tol=1e-06;, score=-109350.489 total time=   0.7s
[CV 4/5] END max_iter=400, n_clusters=4, n_init=1, tol=1e-06;, score=-115035.099 total time=   0.7s
[CV 5/5] END max_iter=400, n_clusters=4, n_init=1, tol=1e-06;, score=-109343.106 total time=   0.4s
[CV 1/5] END max_iter=400, n_clusters=4, n_init=1, tol=1e-05;, score=-103407.973 total time=   0.6s
```

Figure 5: Grid Search KMeans Bow

```
[CV 4/5] END max_iter=300, n_clusters=5, n_init=10, tol=0.0001;, score=-1831.291 total time=  17.6s
[CV 5/5] END max_iter=300, n_clusters=5, n_init=10, tol=0.0001;, score=-1824.257 total time=  20.5s
[CV 1/5] END max_iter=300, n_clusters=5, n_init=15, tol=1e-05;, score=-1839.042 total time=  32.2s
[CV 2/5] END max_iter=300, n_clusters=5, n_init=15, tol=1e-05;, score=-1835.016 total time=  26.7s
[CV 3/5] END max_iter=300, n_clusters=5, n_init=15, tol=1e-05;, score=-1836.636 total time=  30.2s
[CV 4/5] END max_iter=300, n_clusters=5, n_init=15, tol=1e-05;, score=-1831.348 total time=  34.8s
[CV 5/5] END max_iter=300, n_clusters=5, n_init=15, tol=1e-05;, score=-1824.028 total time=  29.3s
[CV 1/5] END max_iter=300, n_clusters=5, n_init=15, tol=0.0001;, score=-1845.335 total time=  34.3s
[CV 2/5] END max_iter=300, n_clusters=5, n_init=15, tol=0.0001;, score=-1842.225 total time=  29.8s
[CV 3/5] END max_iter=300, n_clusters=5, n_init=15, tol=0.0001;, score=-1836.685 total time=  29.1s
[CV 4/5] END max_iter=300, n_clusters=5, n_init=15, tol=0.0001;, score=-1831.330 total time=  29.1s
[CV 5/5] END max_iter=300, n_clusters=5, n_init=15, tol=0.0001;, score=-1824.051 total time=  28.5s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=1, tol=1e-05;, score=-1838.580 total time=   3.3s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=1, tol=1e-05;, score=-1828.696 total time=   3.5s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=1, tol=1e-05;, score=-1833.073 total time=   3.3s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=1, tol=1e-05;, score=-1830.520 total time=   1.6s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=1, tol=1e-05;, score=-1830.180 total time=   5.5s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=1, tol=0.0001;, score=-1837.927 total time=   1.5s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=1, tol=0.0001;, score=-1828.712 total time=   1.6s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=1, tol=0.0001;, score=-1824.069 total time=   2.4s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=1, tol=0.0001;, score=-1827.587 total time=   1.5s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=1, tol=0.0001;, score=-1817.018 total time=   2.0s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=5, tol=1e-05;, score=-1837.941 total time=  11.0s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=5, tol=1e-05;, score=-1828.703 total time=  13.3s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=5, tol=1e-05;, score=-1835.060 total time=  11.8s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=5, tol=1e-05;, score=-1828.614 total time=  13.2s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=5, tol=1e-05;, score=-1815.023 total time=  10.3s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-1837.930 total time=  10.4s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-1828.715 total time=  11.5s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-1825.569 total time=  10.4s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-1828.617 total time=  12.7s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=5, tol=0.0001;, score=-1819.439 total time=   9.7s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-1837.946 total time=  25.1s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-1828.687 total time=  19.8s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-1825.675 total time=  22.3s
[CV 4/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-1828.596 total time=  22.7s
[CV 5/5] END max_iter=300, n_clusters=6, n_init=10, tol=1e-05;, score=-1819.804 total time=  21.0s
[CV 1/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-1837.979 total time=  26.8s
[CV 2/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-1828.668 total time=  21.8s
[CV 3/5] END max_iter=300, n_clusters=6, n_init=10, tol=0.0001;, score=-1835.093 total time=  26.3s
```

Figure 6: Grid Search Kmeans TF-IDF

5 Agglomerative Clustering

5.1 Definition of the model

Agglomerative Clustering is a part of hierarchical clustering. It has 4 types of linkage criteria: ward, complete, average and single. The hierarchy is represented like a dendogram and it it created with a bottom up approach.

5.2 Silhouette Coefficient

The Silhouette Coefficient is a metric used in order to check the goodness of a clustering technique. This result has a value between 1 and -1, where 1 means a perfect cluster, 0 is neutral and -1 is a warning sign.

I have computed just two Silhouette Coefficients for the Agglomerative method when using BoW because it lasts a long time to process, in Table 1 we can see the results.

Table 1: Table 1

Metric	Linkage	Silhouette Coefficient
euclidean	ward	0.1246
manhattan	average	0.7015

I have computed just two Silhouette Coefficients for the Agglomerative method when using TF-IDF because it lasts a long time to process, in Table 2 we can see the results.

Table 2: Table 2

Metric	Linkage	Silhouette Coefficient
euclidean	ward	0.0074
manhattan	average	0.0074

6 Comparison

I have computed the cluster visualization for a number of trails using both K-Means and Agglomerative clustering. In this comparison section I will only present the results given on 5 cluster distributions.

In Figure 7 and Figure 8 we can see the representation of K-Means on BoW features.

In Figure 9 and Figure 10 we can see the representation of K-Means on TF-IDF features.

In Figure 11 and Figure 12 we can see the representation of Agglomerative on BoW features.

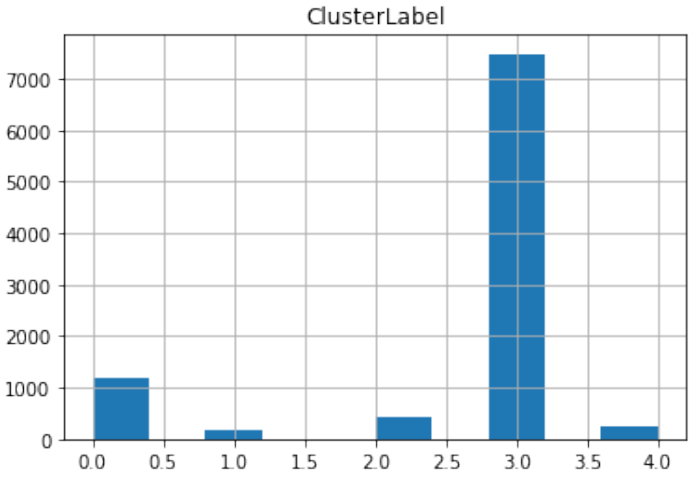

Figure 7: K-Means BoW Clusters Histogram

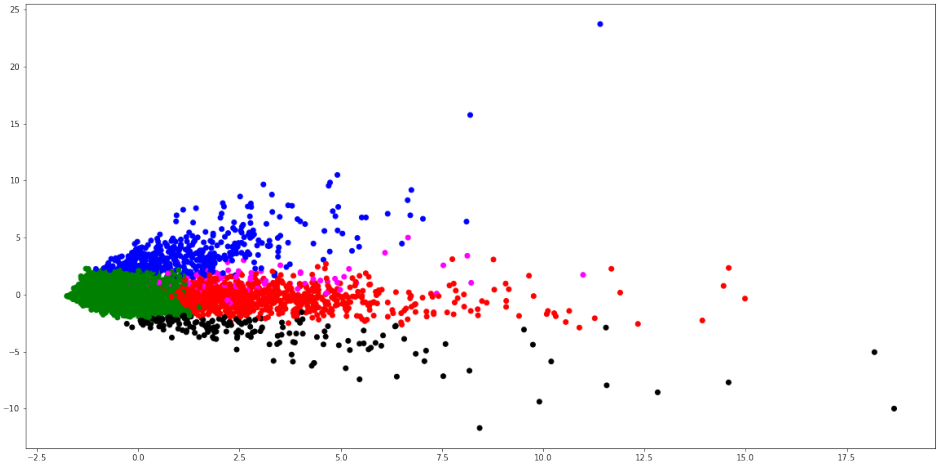

Figure 8: K-Means BoW Clusters Scatter

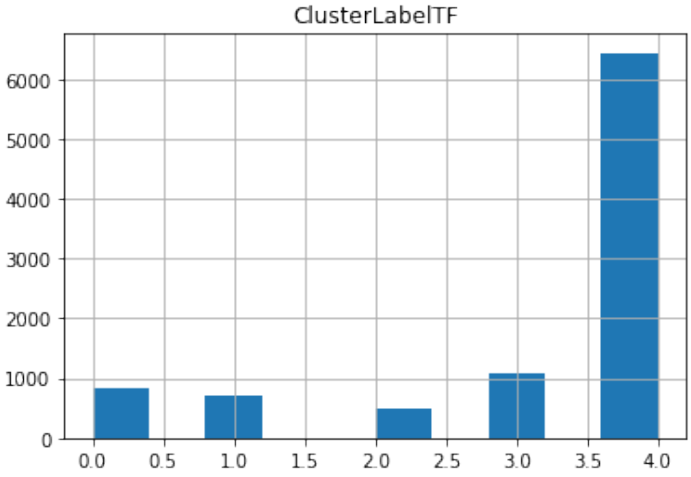

Figure 9: K-Means TF-IDF Clusters Histogram

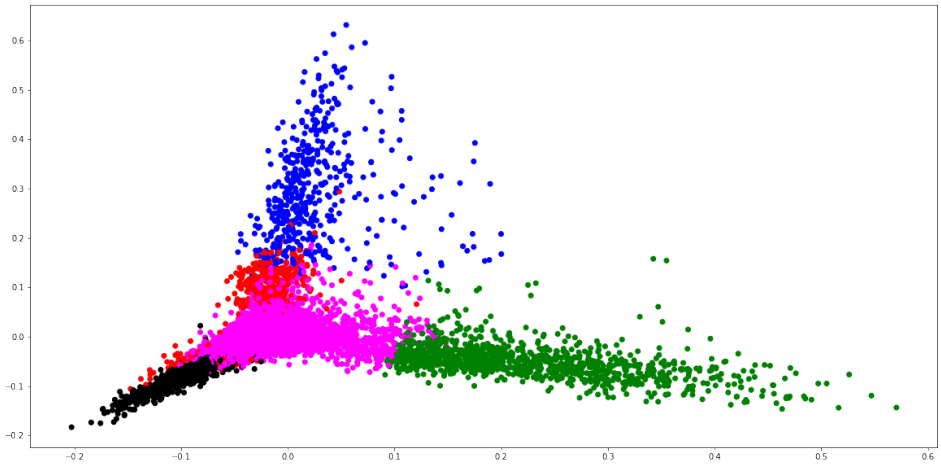

Figure 10: K-Means TF-IDF Clusters Scatter

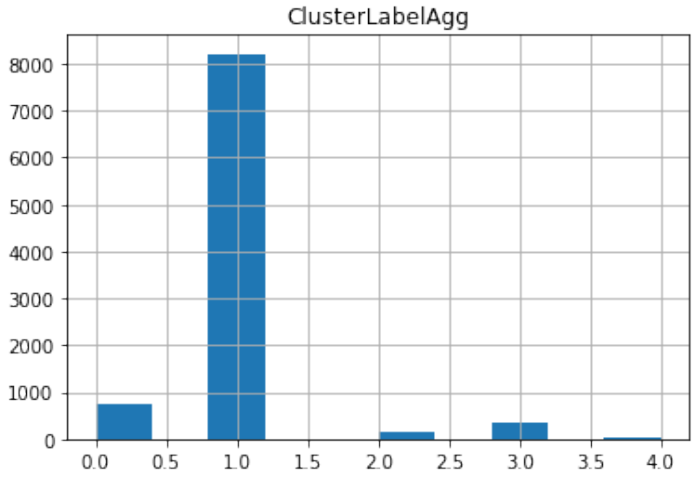

Figure 11: Agglomerative BoW Clusters Histogram

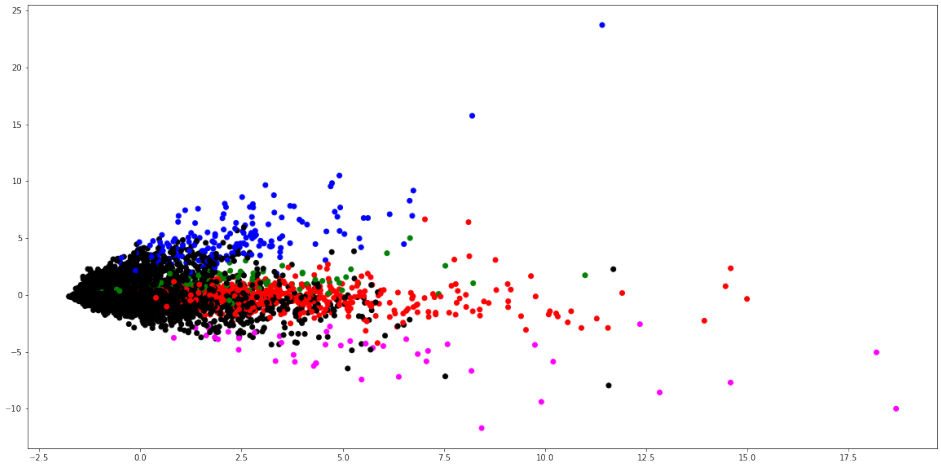

Figure 12: Agglomerative BoW Clusters Scatter

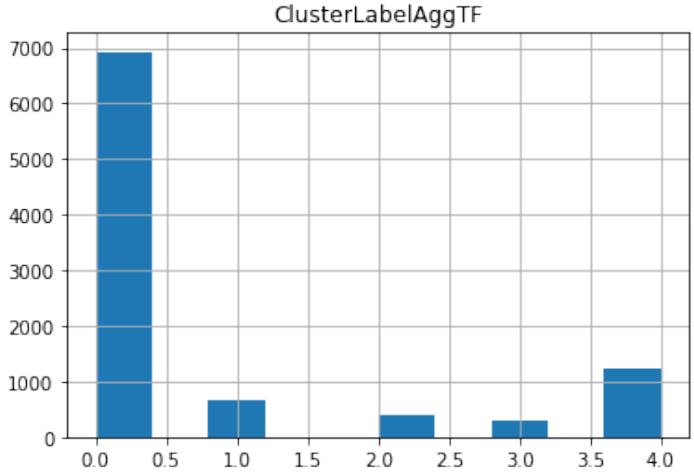

Figure 13: Agglomerative TF-IDF Clusters Histogram

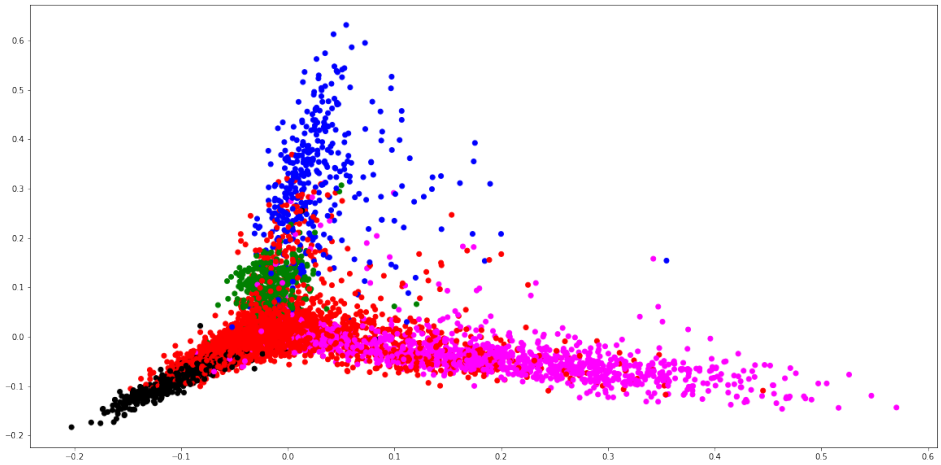

Figure 14: Agglomerative TF-IDF Clusters Scatter

In Figure 12 and Figure 14 we can see the representation of Agglomerative on TF-IDF features.

We can observe from all of these figures, similarly to the score representation, that there is a cluster with much more reviews than any other. Also, we can see a tendency to form weLl defined clusters that is more obvious in the K-Means model than the Agglomerative one.

For the purpose of comparison, I have computed a random distribution of clusters for both types of features, which can be seen in Figure 15 and Figure 16. I have also compiled a random distribution histogram of clusters: Figure 17.

It is clear that the random distribution of clusters is doing a worse job at splitting the data into clusters than the unsupervised methods. On this set of data, on the types of features that I used for unsupervised learning, I also used a supervised learning model: SVM. The accuracy I got for SVM was 0.62. I didn't get a chance to compare the way the clusters scatter with the supervised method, but from the experiments that I've done on the tests set, I believe the data is better split by the SVM method.

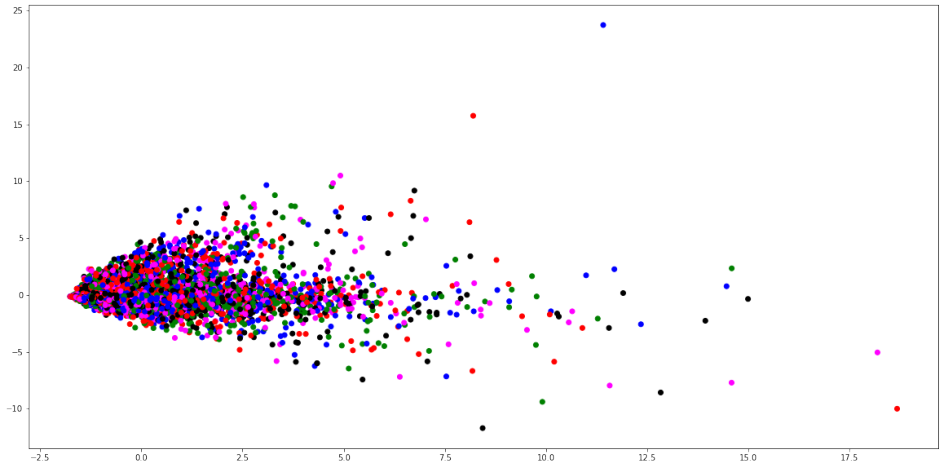

Figure 15: Random Clusters BoW Scatter

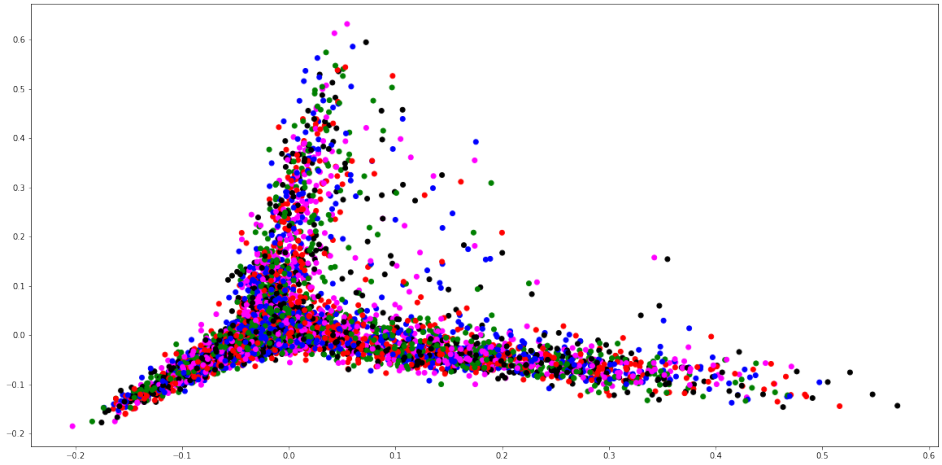

Figure 16: Random Clusters TF-IDF Scatter

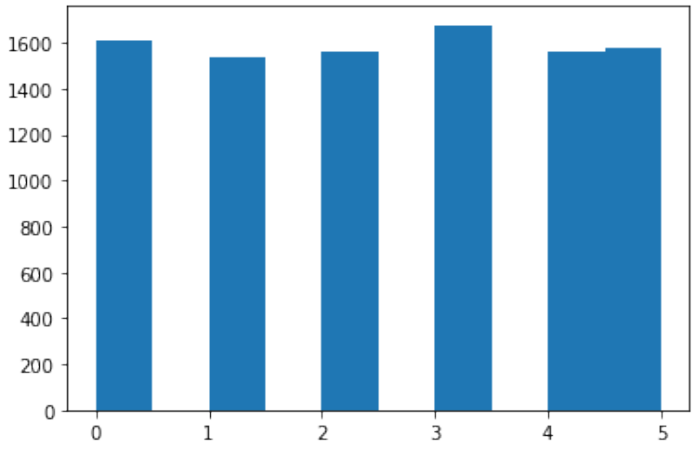

Figure 17: Random Distribution Histogram

7 Conclusion

In conclusion, working with unsupervised learning models can be a challenge, but it is also an interesting topic to perfect. Because of its difficulties, you have to take great care when processing the data and when choosing the features, so that you are using your best options.

As future developments for my models, I would choose to try Stemming instead of Lemmatization and I would consider other types of features as well, for example Word2Vec. Besides this, I would try to do more fine-tuning for the Agglomerative Clustering parameters. Another idea that I have would be transforming the data into PCA before running the K-Means model.