Text Mining Project

Text Analysis of Amazon reviews: clustering, summarization and sentiment analysis

Edoardo Attilio 869272, Luca Duci 851174, Alfio Leanza 910235

Table of contents

Introduction and dataset description

The Amazon Fine Food Reviews dataset, containing over 500,000 customer reviews, provides a rich resource for exploring advanced text mining techniques. Each review includes a numerical score (ranging from 1 to 5), a detailed textual description, and metadata such as user and product identifiers.

This project has three primary objectives: clustering reviews based on their textual content to uncover patterns of customer sentiment and behavior, generating concise summaries to enhance the accessibility of information, and analyzing the sentiment polarity of the reviews. By clustering, we aim to group similar reviews, identifying common themes or shared sentiments among customers. Summarization, in turn, condenses detailed feedback into meaningful highlights, offering businesses a quick and effective way to understand customer opinions. Sentiment analysis allows to quickly understand the (negative, positive or neutral) emotions related to a text. Together, these approaches provide actionable insights that can inform product development, customer engagement strategies, and overall decision-making in the e-commerce sector.

An initial exploration of the dataset revealed several characteristics. The distribution of review scores is skewed toward positive ratings, with the majority falling within the 4 to 5-star range, reflecting high overall customer satisfaction (*Fig. 1*).

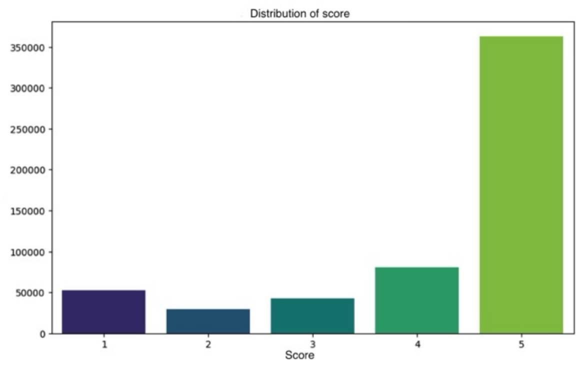


Fig. 1 Histogram showing the number of reviews for each category of the "Score" variable.

Review lengths vary widely, encompassing both brief comments and extensive feedback (*Fig. 2*). Additionally, some products and users stand out for their high levels of activity, suggesting either exceptional popularity or consistent engagement.

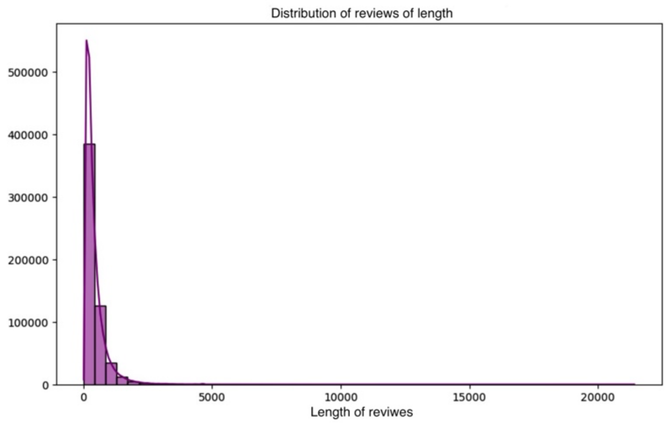


Fig. 2 Distribution of length reviews.

Pre-Processing

Data preprocessing is a crucial step in preparing the dataset for analysis, ensuring its quality and compatibility with the intended tasks of clustering and summarization. We decided to remove irrelevant columns for our tasks such as *ProfileName*, *HelpfulnessNumerator*, *HelpfulnessDenominator* and *Time*, as they do not contribute directly to textual analysis. Additionally, rows containing missing values in the text or summary fields were identified and dropped. This decision was justified by the minimal impact on the dataset's overall size, given the low number of missing entries.

Clustering Pre-Processing

For clustering, the text data underwent extensive cleaning to ensure uniformity and eliminate noise:

- Lowercasing: all text was converted to lowercase to remove case-related inconsistencies.
- Noise Removal: non-alphabetic characters and numbers were stripped from the text.
- Tokenization: each review was broken into individual words for further processing.
- Stop Words Removal: common but non-informative words such as "*and*" "*the*" and "*is*" were excluded using NLTK's English stopword list.

- <u>Lemmatization</u>: words were reduced to their base forms (e.g., "*running*" to "*run*") using WordNet's lemmatizer.

The cleaned text was then saved as a new column, *Cleaned_Text_Clustering* and exported for clustering purposes. This preprocessed data will enable more meaningful grouping of reviews based on their content.

Example of Clustering Pre-processing:

<u>Original text</u>: «*I have bought several of the Vitality canned dog food products and have found them all to be of good quality. The product looks more like a stew than a processed meat and it smells better. My Labrador is finicky and she appreciates this product better than most.*»

<u>After Pre-Processing</u>: «*bought several vitality canned dog food product found good quality product look like stew processed meat smell better labrador finicky appreciates product better*»

Summarization Pre-processing

The summarization task required a more nuanced approach, as both the review text and its corresponding summary were processed. Pre-processing steps included:

- <u>Character Filtering</u>: retaining only alphanumeric characters and punctuation necessary for sentence coherence.
- <u>Whitespace Normalization</u>: extra spaces were removed to maintain consistency.

Similar to the clustering task, missing summaries were minimal and thus dropped without significant loss of data. Pre-processed text and summaries were then saved as new columns, *Cleaned_Text_Summarization* and *Cleaned_Summary*. These refined inputs ensure compatibility with summarization algorithms while preserving essential content.

The preprocessing steps significantly enhanced the dataset's usability, ensuring its alignment with the project's clustering and summarization goals. By transforming raw data into a clean and structured format, we have laid a solid foundation for uncovering valuable insights.

Text clustering

The first task we performed was text clustering, which aims to group objects (in this case, reviews) into distinct groups/clusters. The goal is to maximize the similarity of texts within the same cluster while minimizing similarity between texts in different clusters. Various clustering techniques can be applied; we tried to work with both flat and hierarchical approaches, specifically K-means and agglomerative methods.

To apply clustering in an efficient way, we had to reduce the number of reviews to consider: working with more than one half-million texts could become tedious and extremely long and, for this reason, we worked with a small amount of them. To obtain the new smaller dataset, we used a stratified proportional sampling, using the variable *Score* to stratify. A "stratified proportional sampling" enables the creation of a smaller dataset that preserves the original proportions of the values of a specific variable from the full dataset. In this way the new sampled dataset contains, for each level of *Score*, the same proportions of reviews as in the original dataset. The values of this variable refer to the number of stars given by the users to the products and has five levels: 1, 2, 3, 4, and 5 and, the higher the value, the more positive the review. We sampled 1% of the original dataset and kept only 5684 reviews.

Finally, before applying clustering, using the *TF-IDF* weighting system, we created the vectorial representation of the texts. The Term Frequency-Inverse Document Frequency (*TF-IDF*) assigns importance weights to terms within a document based on their frequency of occurrence and their rarity across the dataset. *TF-IDF* vectorization ensures that common words are down-weighted while unique terms are emphasized. In this context: Term Frequency (*TF*) quantifies the occurrence of a term within a document; Inverse Document Frequency (*IDF*) measures the rarity of the term across the corpus. The incidence matrix contains 5684 (which are the reviews) and 17394, the number of words (types) found in the texts.

K-means

The *K-means* clustering algorithm is a flat technique that partitions data into *K* clusters by minimizing the sum of squared distances between data points (in our case, the vectors which represent the reviews) and their cluster centroids. This method requires as input not only the data to cluster but also the number of groups we want to create. Since it is impossible to know a priori the optimal number of clusters to generate, we should try different values of *K* and then choose the best one. To do so, we used the "elbow method": by plotting how distortion changes as the number of clusters increases, we selected the value corresponding to the "elbow" of the curve (where the curve "visibly bends"). To apply the elbow method, we used the `yellowbrick` library, which considers as "distortion" the "sum of squared distances from each point to its assigned center". At the same time, though, it is possible to use other metrics, such as the "silhouette" (which calculates the mean Silhouette Coefficient of all samples). In *Fig. 3* and *Fig. 4* it is possible to see the outputs we obtained:

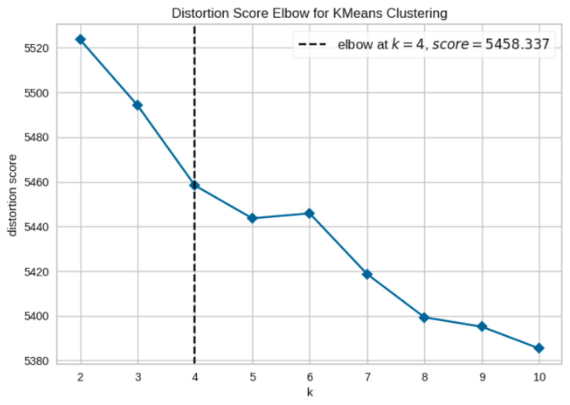


Fig. 3 *Score of distortion across different values of the parameter K.*

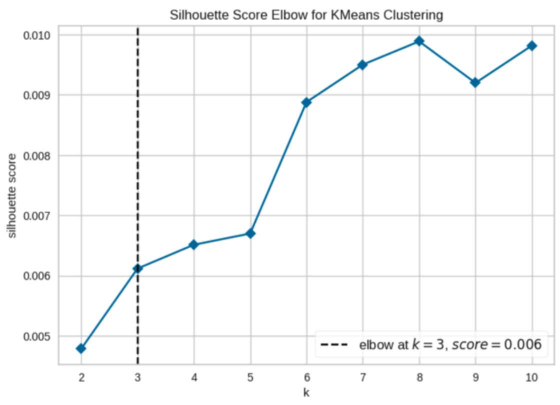


Fig. 4 *Score of silhouette across different values of the parameter K.*

According to the "distortion" metric, the elbow is in $K = 4$, while instead for the "silhouette" is in $K = 3$; at the same time, though, if we look at the right graph, it seems that there is a strong difference of silhouette score also between $K = 5$ and $K = 6$. As a first valuation, we decided to use $K = 4$.

After applying the *K-means* algorithm with 4 clusters, we analyzed the results. Firstly, we created a plot of the clusters using a dimension reduction method: we applied PCA, first with two components (*Fig. 5*) and then with three (*Fig. 6*).

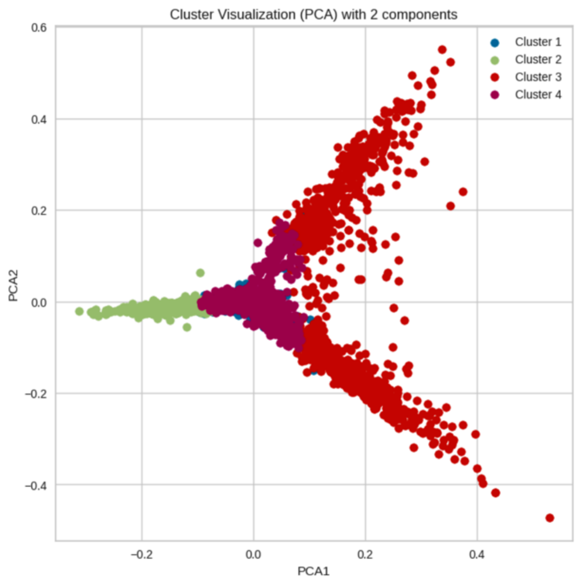


Fig. 5 *Data points plotted using PCA (2 components); different color refers to different cluster.*

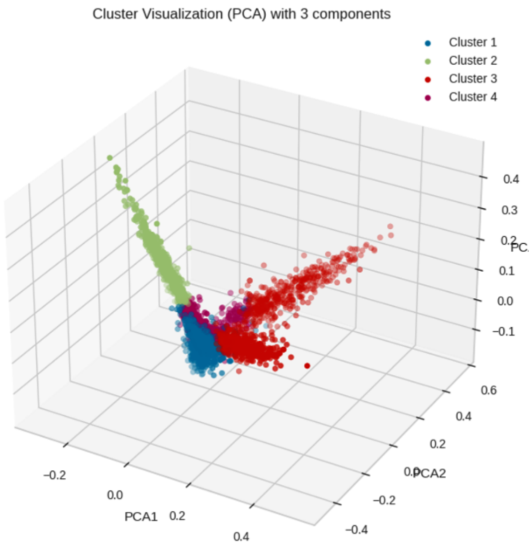


Fig. 6 *Data points plotted using PCA (3 components); different color refers to different cluster.*

Looking at *Fig. 5* (the one with two PCs) it seems that five might be a better number of clusters, since the points of cluster "3" appear quite far from each other; nevertheless, points from clusters "1" and "4" almost completely overlap. Also looking at *Fig. 6* (the one with three PCs), it is possible to see those points from clusters "1" and "4" are very close.

The number of reviews for each cluster is in *Tab. 1*.

Cluster	Num. of docs
1	797
2	621
3	938
4	3328

Tab. 1 *Number of reviews for each cluster*

Cluster "4" is much more populated than the other three. Finally, we created the word clouds for each cluster, removing the word "product" from it: all the reviews refer to a specific Amazon product and thus this word represents all the created clusters.

For cluster "1" (*Fig. 7*). It seems that this cluster contains words related to <u>sweet food</u> and <u>snacks</u>: *chocolate*, *snack*, *chip* (in the sense of chips to eat), *peanut*, *almond*, *butter*, *bar* (in the sense of snack bar), *sweet*, *sugar*, *fat*, are some of the most frequent words. Among these frequent words we can also find *cooky* which does not mean anything in English; after analyzing the original texts we discovered that it is the word "cookie" that, after pre-processing, becomes *cooky*.

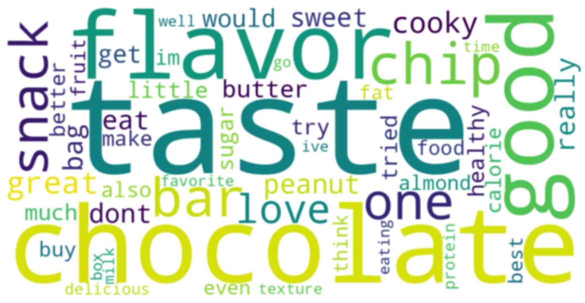


Fig. 7 *Word clouds of cluster "1".*

For cluster "2" (*Fig. 8*):

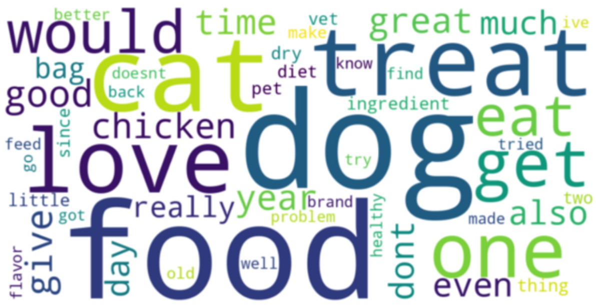


Fig. 8 Word clouds of cluster "2".

It seems that this cluster contains words related to <u>animal food</u>: *dog*, *cat*, *vet*, *treat*, *chicken* and *pet* are some of the most common words.

For cluster "3" (*Fig. 9*):

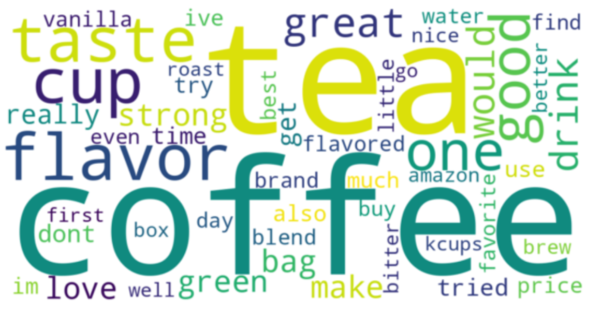


Fig. 9 Word clouds of cluster "3".

It seems that this cluster contains words related to <u>beverages</u>, mainly tea and coffee: *tea*, *coffee*, *cup*, *drink*, *green* (in the sense of green tea), *water*, *brew*, *vanilla* and *blend* are some of the most common words.

Finally, cluster "4" (*Fig. 10*). In this case it does not seem possible to recognize any precise topic. Some of the most common words seem to refer to positive reviews: *good*, *great* and *love*.

To conclude, we can say that the five clusters were likely created according to the type of product sold. Then, we asked ourselves if there was any kind of relation between the score of the review and the cluster division: did the K-mean algorithm divide the reviews according to their *Score*? In *Tab. 2* it is possible to see the average of the variable *Score* for each cluster.

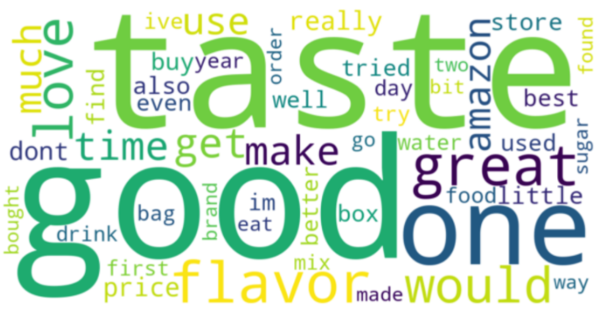


Fig. 10 Word clouds of cluster "3".

Cluster	Avg. score
1	4.35
2	4.22
3	4.18
4	4.14

Tab. 2 The mean values of the variable "Score" for each cluster.

The means all seem similar to each other and thus we can conclude that the clustering algorithm probably did not consider the score given by users to the products but only the type of product discussed in the review.

Since we do not have an external criterion to compare our clustering results to, we used two internal indices to evaluate the results. The *silhouette index* is equal to 0.006: not a great value but at least is not negative (which would indicate that, on average, data points are closer to points in other clusters than to points in their own cluster). The Calinski-Harabasz score is quite low since it is equal to 28.

Agglomerative clustering

Agglomerative clustering is a hierarchical technique which works as follows: from a situation where the number of clusters is equal to the number of data points that we have, the algorithm, step by step, merges two different clusters and it stops where all the points are in the same group. The steps of a hierarchical clustering can be visualized using a dendrogram (*Fig. 11*); "Ward" method used for linkage. As we can see from the figure, the number of suggested clusters is five: for this reason, we applied an agglomerative clustering algorithm (using "Ward" as linking method and Euclidean distances) and created five groups. The final silhouette value in this case is equal to 0.004 while the Calinski-Harabasz score is 24.9: both values appear lower than the ones obtained before and thus we can conclude that this clustering technique performs slightly worse than the previous one.

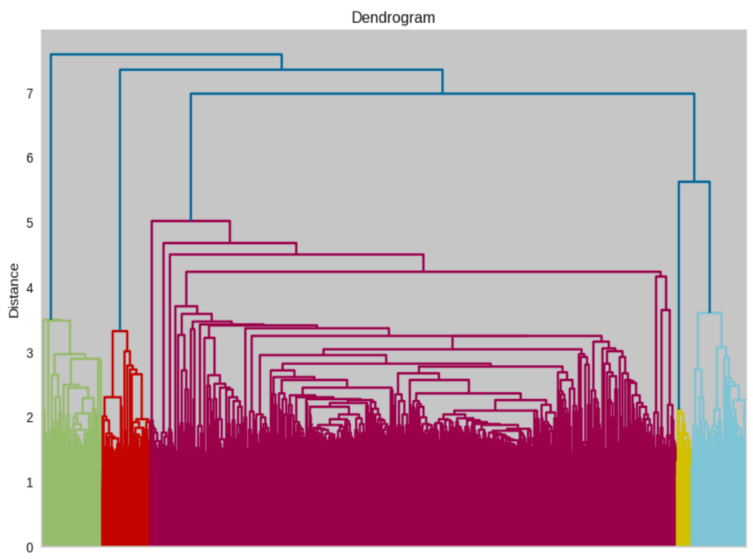


Fig. 11 Dendrogram of the agglomerative clustering using "Ward" linkage.

Improvement for K-means

The results of the K-means algorithm not only depend on the choice of the number of clusters *K*, but it also depends on the (random) selection of the first *K* vectors that are used to initialize the procedure. We could try to improve the algorithm by choosing as the first initializing vectors the centroids of another clustering procedure, such as the agglomerative one (which does not even require a pre-defined number of clusters as input). So, we tried to apply the K-means algorithm using the five centroids obtained with the agglomerative method as starting vectors and with $K = 5$. The plot of the clusters using a dimension reduction method (PCA with two components) can be seen in *Fig. 12*.

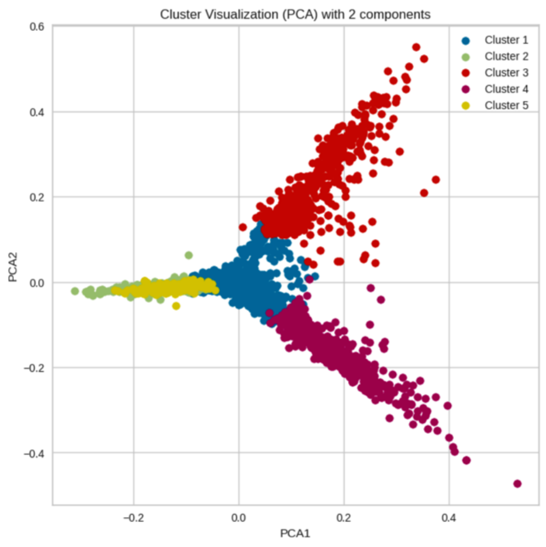


Fig. 12 Data points plotted using PCA (2 components); different color refers to different cluster.

Those far located points that before were placed in the same cluster are now divided into two different ones (cluster "3" and "4"); now, considering only two PCs, it seems that points from cluster "2" and "5" overlap.

The silhouette measure is now equal to 0.008 and the Calinski-Harabasz score is 32: both measures are the highest so far, it appears that a better choice of the initial centroids and the use of five clusters can actually improve, even if slightly, the clustering performance. Looking at the word clouds it is possible to recognize the main topics of the five clusters (*Fig. 13*): dog food (cluster "2"), tea and drinks (cluster "3"), coffee and drinks (cluster "4") and cat food (cluster "5"); for cluster "1", instead, it is not possible to recognize any particular topic (like in the case of cluster "4" in the first K-means clustering we tried).

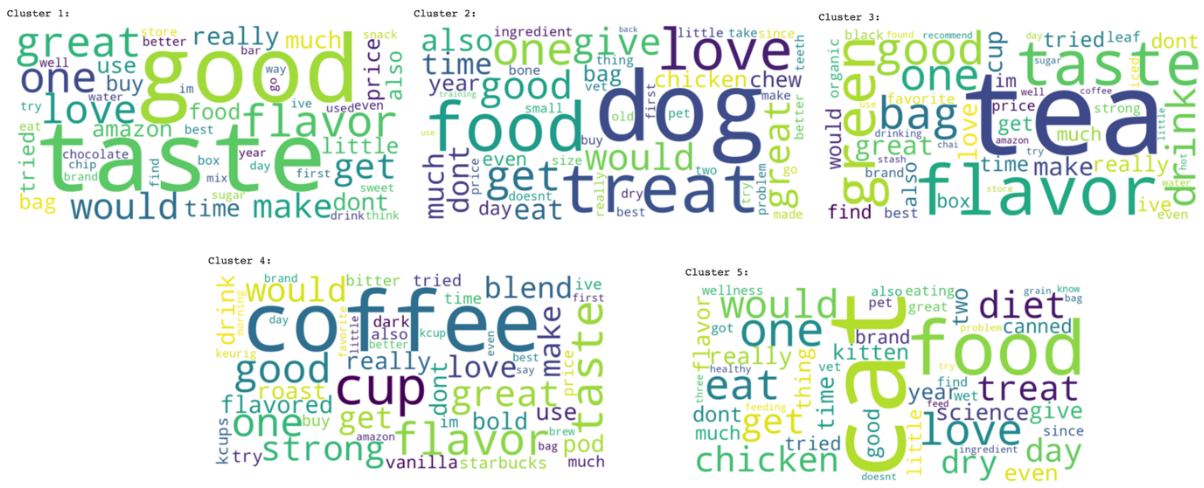


Fig. 13 *Word clouds of the five clusters*

In this section, we discussed text clustering: we applied a couple of clustering techniques to our data and analyzed the results. The *K-means* algorithm performs slightly better than the hierarchical clustering; moreover, a not-random vector initialization increases the performance of the algorithm. It is possible to recognize different topics in different clusters.

Text summarization

The second task we computed was text summarization, which has the goal of enabling efficient extraction of meaningful content from textual data. This phase implements extractive summarization techniques to condense text into representative summaries. The approach identifies important sentences directly from the input text based on statistical and graph-theoretic methods.

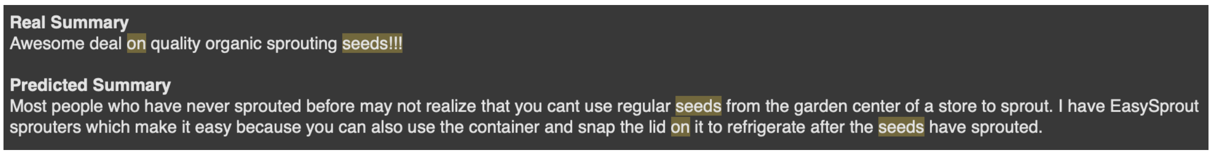
The methodology begins with dataset preparation, where a sample of 1% (5,684 rows) from the dataset is used to test the summarization approach. The textual data, stored in the *'Cleaned_Text_Summarization'* column, is preprocessed to remove noise and ensure consistency. To get summarization the texts provided underwent some steps contained in the function "*summarize_doc*":

- Feature Representation of texts (TF-IDF).

- Sentence Similarity Computation: a cosine similarity matrix was constructed to quantify the relationships between sentence vectors. Cosine similarity measures the cosine of the angle between two vectors in a multi-dimensional space. This step is critical for constructing a weighted graph where nodes represent sentences and edges capture their similarity.

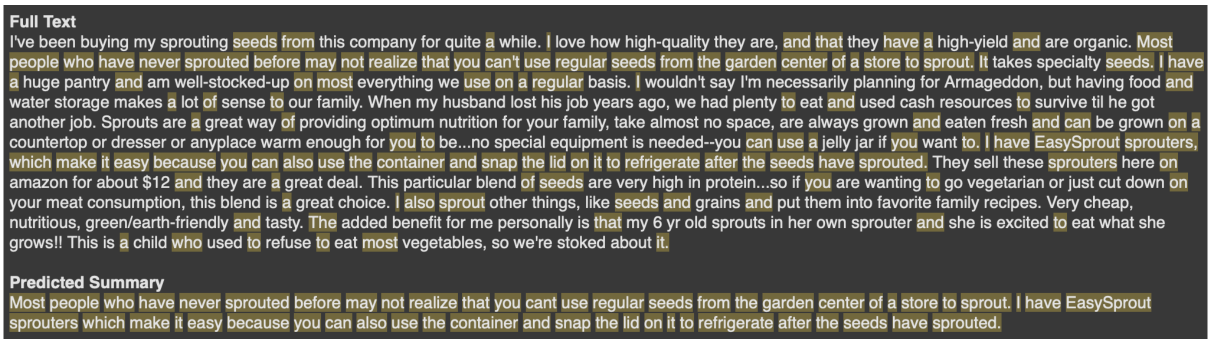
- Graph-Based Ranking (PageRank Algorithm): the PageRank algorithm, originally designed for ranking web pages, was adapted for ranking sentences based on their interconnectedness within the similarity graph. Sentences with higher connectivity to other high-scoring sentences were assigned greater importance. The damping factor (*d*), set to 0.85, balances the influence of connectedness and randomness in the graph traversal.
- Summary Extraction: The top-ranked sentences, as determined by their PageRank scores, were extracted to form the summary. The number of sentences (*k*) included in the summary was predefined based on the desired compression ratio.

After computing the function "*summarize_doc*", thanks to another function "*display_string_matching*", we compared firstly the real summary against the predicted:


Real Summary
Awesome deal on quality organic sprouting seeds!!!

Predicted Summary
Most people who have never sprouted before may not realize that you cant use regular seeds from the garden center of a store to sprout. I have EasySprout sprouters which make it easy because you can also use the container and snap the lid on it to refrigerate after the seeds have sprouted.

and then the original text always against the predicted summary:


Full Text
I've been buying my sprouting seeds from this company for quite a while. I love how high-quality they are, and that they have a high-yield and are organic. Most people who have never sprouted before may not realize that you can't use regular seeds from the garden center of a store to sprout. It takes specialty seeds. I have a huge pantry and am well-stocked-up on most everything we use on a regular basis. I wouldn't say I'm necessarily planning for Armageddon, but having food and water storage makes a lot of sense to our family. When my husband lost his job years ago, we had plenty to eat and used cash resources to survive til he got another job. Sprouts are a great way of providing optimum nutrition for your family, take almost no space, are always grown and eaten fresh and can be grown on a countertop or dresser or anyplace warm enough for you to be...no special equipment is needed--you can use a jelly jar if you want to. I have EasySprout sprouters, which make it easy because you can also use the container and snap the lid on it to refrigerate after the seeds have sprouted. They sell these sprouters here on amazon for about $12 and they are a great deal. This particular blend of seeds are very high in protein...so if you are wanting to go vegetarian or just cut down on your meat consumption, this blend is a great choice. I also sprout other things, like seeds and grains and put them into favorite family recipes. Very cheap, nutritious, green/earth-friendly and tasty. The added benefit for me personally is that my 6 yr old sprouts in her own sprouter and she is excited to eat what she grows!! This is a child who used to refuse to eat most vegetables, so we're stoked about it.

Predicted Summary
Most people who have never sprouted before may not realize that you cant use regular seeds from the garden center of a store to sprout. I have EasySprout sprouters which make it easy because you can also use the container and snap the lid on it to refrigerate after the seeds have sprouted.

Visualizing that we can conclude that the summary taken from the original dataset does not implement an extractive method, due to the new words that original text does not contain. About our predicted summary we can say, at first, that the model implemented is able to synthetize a text.

Evaluation

To quantitatively measure the quality of the summaries, the following metrics were calculated using the `rouge` library:

- ROUGE-1: measures the number of matching n-gram unigrams between the actual and predicted summaries.
- ROUGUE-2: calculates the overlap of bigrams between the two texts.

- ROUGE-L: evaluates the length of the longest common subsequence (Longest Common Subsequence) as an indicator of structural coherence.

For each pair of summaries, scores were calculated and then aggregated to provide an overall assessment. Precautions were taken to handle cases where one of the summaries was empty, assigning scores of zero to avoid bias.

- Mean ROUGE-1: 0.0521
- Mean ROUGE-2: 0.0104
- Mean ROUGE-L: 0.0497

The results from the evaluation are relatively low because the metrics refer to similarity with the real or initial summary. Having as a model of comparison summaries through methodologies other than the executive summary, we will proceed to make other evaluations, and for this reason we reach these very low results.

Other metrics in terms of executive summary can be the Coverage and the Compression:

- Coverage: measures the amount of relevant content from the original text that was included in the summary. It is calculated as the ratio of the number of words (or phrases) shared between the summary and the original text to the total number of unique words (or phrases) in the original text.
- Compression: measures how concise the summary is compared to the original text. It is calculated as the ratio of the number of words in the summary to the number of words in the original text.

After computing them we had:

- Mean Coverage: 0.58
- Mean Compression: 0.62

So, with a Mean Coverage of 0.58 and a Mean Compression of 0.62, it can be seen that the model generates summaries that tend to retain a significant portion of the main information (about 52 percent of the original keywords or concepts), but at the cost of less incisive synthesis, as the summaries generated are on average 68 percent of the length of the original text.

These values suggest that the model has a good ability to preserve informative content, but is less effective in creating concise and concise summaries. In practical terms, this could mean that the summaries turn out to be detailed, but not short enough to be considered optimal in contexts where synthesis is critical.

Sentiment analysis

As mentioned earlier, the data we analyzed consists of Amazon review: for this reason, we decided to apply a couple of sentiment analysis methods to evaluate their effectiveness on our dataset.

Sentiment analysis is a text analysis task which aims at analyzing the emotions and sentiments from text. It is divided into two primary tasks: emotion recognition and polarity detection. Our focus was on the latter. Polarity detection involves assessing the degree of positivity, negativity, or neutrality in a given text. Like in all the other text mining tasks, also in sentiment analysis many different methods can be used: we decided to focus on two lexicon-based approaches, Afinn and Vader. Lexicon-based methods rely on a predefined "vocabulary" of words where each one of them is associated with a sentiment score (that, as we have said before, it can be either positive, negative or neutral). The final score of a text is given by the combination of the sentiment scores of the words in it. Lexicon-based approaches, in general, are easy to implement and work very fast; at the same time, though, they are not able to analyze the context of a text and thus it is impossible for them to detect irony, sarcasm and particular phrases or idiomatic expressions. At the same time, it makes sense to imagine that more complex usages of the language, that require a deeper understanding of the context, are not used in Amazon products' reviews, where the users tend to be as clear as possible.

To apply sentiment analysis, we have decided to work with different dataset from the one we have used up to this point: since we want to test the capabilities of the two previously mentioned techniques, we thought that it would have made more sense to consider an equal number of reviews for each value of the variable *Score* (and thus applying a disproportionate stratified sample). The values of this variable refer to the number of stars given by the users to the products: for this reason, *Score* could be considered the "reference"/"golden standard". For each level we have collected 2000 reviews for a total of 10000 reviews.

Finally, before applying sentiment, we also created a new rank system, re-coding the *Score* variable in a three-level variable with values "negative" (for reviews with a 1 or 2 score), "neutral" (for reviews with a score equal to 3) and "positive" (for reviews with a 4 or 5 score).

Afinn

Afinn scores words in a [-5; +5] scale according to their polarity, with negative scores for words deemed "negative" and positive scores for those considered "positive". This method does not require any kind of particular text pre-processing steps. After applying the Afinn technique to our data, we can see the distribution of the scores given to the reviews (*Fig. 14*):

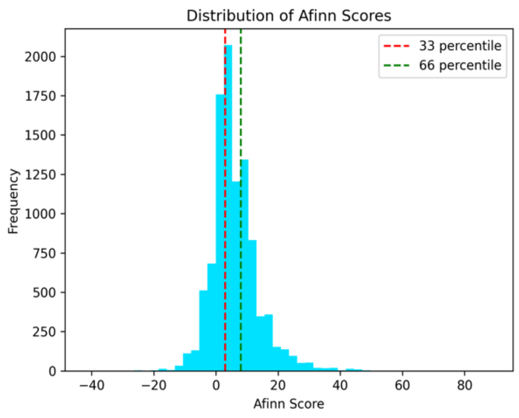


Fig. 14 *Distribution of Afinn scores*

As we can see, the majority of the reviews have a score between -20 and +20. At the same time, though, we can notice that some of them are scored much higher/lower than the others. These reviews are likely the longer ones: longer reviews tend to include more words, which increases the likelihood of containing more scored words, ultimately resulting in a higher overall score of the review. It is not surprising noticing that the top five most positive and negative reviews are all quite long indeed. Moreover, in the graph we plotted the 33rd and the 66th percentiles of the Afinn scores distribution: the 33th percentile is equal to 3, while the other is equal to 8. It is interesting to note that both of these values are positive: in fact, in general, most of the given scores are positive. We then computed the average Afinn score stratifying for the *Score* variable. The results are in *Tab. 3*

Score	**Avg. Afinn**
1	1.44
2	4.56
3	6.49
4	8.85
5	8.91

Tab. 3 *The average Afinn scores for each level of the variable "Score".*

As we can see, as the values of *Score* grow, the average values of the Afinn score do the same; at the same time, though, we can see that the growth of the Afinn scores is not constant among the values of *Score*: the difference between the average Afinn referring to *Score* = 1 and the average Afinn referring to *Score* = 2 is much wider than the difference between the average Afinn referring to *Score* = 4 and the average Afinn referring to *Score* = 5, which are instead quite similar. To better understand the results, we plotted the scores obtained using the Afinn method in the five classes of *Score* (*Fig. 15*):

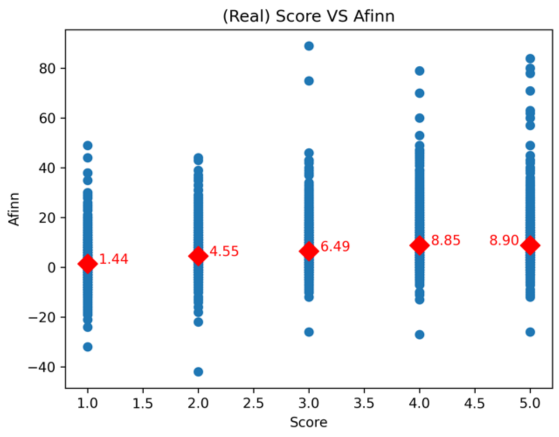


Fig. 15 *Afinn scores for each level of Score.*

The red dots represent the average value of that class. The plot shows that, while the average values differ between the various *Score* levels, the overall range of Afinn values remains relatively consistent. However, for four- and five-star reviews, some show notably higher Afinn scores. It is interesting to notice that the review with the highest Afinn score is actually a 3-stars review while the one with the lowest Afinn score review has 2 stars. Then, similarly to what we did before for the *Score* variable, we re-coded the obtained Afinn scores in a three-level discrete variable. To do this, we chose two different couples of cut-offs. In the first case, we used -3 and +3 as values: all reviews with an Afinn score lower than -3 were classified as "negative", the ones with a score greater than +3 were classified as "positive", the remaining ones as "neutral". In the second case we instead used the two previously computed percentiles: all reviews with an Afinn score lower than +3 were classified as "negative", the ones with a score greater than +8 were classified as “positive”, the remaining ones as “neutral”. Finally, we compared these inferred labels with the “real” scores (the ones of the *Score* variable). Results can be seen in *Tab. 4* and *Tab. 5*.

Afinn Var. / ***Score Var.***	**Negative**	**Neutral**	**Positive**
Negative	651	1431	1918
Neutral	88	499	1413
Positive	76	511	3413

Tab. 4 *Levels of the variable “Score” compared to the ones inferred by Afinn (Afinn variable), using -3 and +3 as cut-offs.*

Afinn Var. / ***Score Var.***	**Negative**	**Neutral**	**Positive**
Negative	2382	809	809
Neutral	758	547	695
Positive	869	1169	1962

Tab. 5 *Levels of the variable “Score” compared to the ones inferred by Afinn (Afinn variable), using 33rd and 66th percentiles as cut-offs.*

The tables show that using -3 and +3 as cut-offs results in significantly more reviews being classified as positive compared to when the cut-offs are set at +3 and +8: this happens since the majority of the reviews have a positive Afinn score. Both couples of cut-offs classify many reviews as “neutral” even though their “real” class is not. This problem is probably due to the fact that the number of reviews which “real” class is “neutral” are 2000, while the number of reviews with a “real positive” or “real negative” class is 4000. Finally, we computed a sort of “accuracy” measure as the number of reviews that are classified in the same way both by the *Score* variable and the “Afinn” variable divided by the total number of reviews: in the case of the [-3, +3] cut-offs, the accuracy is 0.456 while using the other cut-offs the accuracy is 0.489: it seems that using the two percentiles allow us to obtain slightly better results.

Vader

Vader is another lexicon-based approach designed to address the limitations of Afinn. Unlike Afinn, Vader accounts for more linguistic features, such as negations, intensifiers, capitalized words, and more. Additionally, Vader provides a normalized sentiment score ranging from -1 to 1. After applying Vader to our data, we can see the distribution of the scores (*Fig. 16*). Differently from the one provided by Afinn, the distribution of the Vader scores is highly negatively skewed: the majority of the scores is positive and higher than the 0.5. As before, we also plotted the 33rd and 66th percentiles, which are 0.42 and 0.87 respectively. In general, we can conclude that Vader tends to classify our reviews with higher values. We then computed the average Vader score stratifying for the *Score* variable: results are in *Tab. 6*.

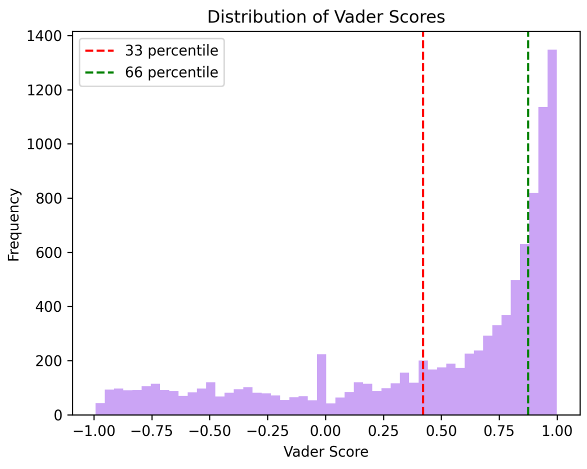


Fig. 16 *Distribution of Vader scores.*

Score	Avg. Afinn
1	0.05
2	0.29
3	0.50
4	0.73
5	0.79

Tab. 6 *The average Vader scores for each level of the variable "Score".*

As expected, as the values of the *Score* grow, the average Vader scores grow too. At the same time, though, as we had noticed before in the case of Afinn, the growth of the Vader scores is not constant among the values of *Score*: the difference between the average Vader referring to *Score* = 1 and the average Vader referring to *Score* = 2 is much wider than the difference between the average Vader referring to *Score* = 4 and the average Vader referring to *Score* = 5, which are instead very close to each other. It is also not surprising that all averages are positive. We then plotted the scores of the Afinn in the five classes of *Score* (*Fig. 17*):

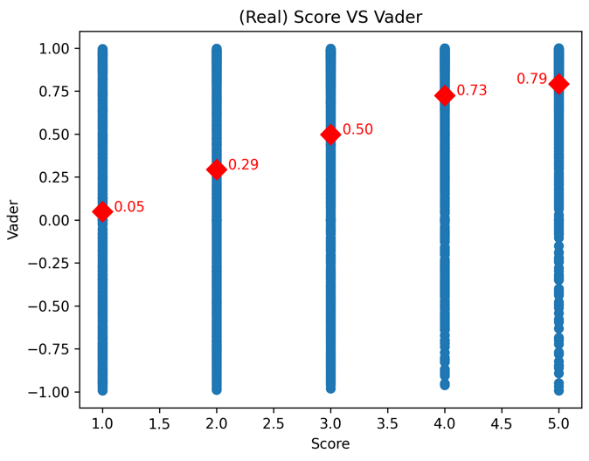


Fig. 17 *Vader scores for each level of Score.*

The red dots represent the average score. As before, also in this case we can notice that, despite the averages of Vader in the five classes of *Score* being quite different, the overall range of Vader values remains relatively consistent.

To conclude, also for the Vader results, we re-coded the values in a three-level discrete variable. Also in this case, we chose two different couples of cut-offs. In the first case, we used -0.05 and +0.05 as values: all reviews with a Vader score lower than -0.05 were classified as "negative", the ones with a score greater than +0.05 were classified as "positive", the remaining ones as "neutral". In the second case we instead used the two percentiles: all reviews with a Vader score lower than 0.42 were classified as "negative", the ones with a score greater than +0.87 were classified as "positive", the remaining ones as "neutral". Finally, we compared these inferred labels with the "real" ones (from the *Score* variable): results can be seen in *Tab. 7* and *Tab. 8*.

Afinn Var. / ***Score Var.***	**Negative**	**Neutral**	**Positive**
Negative	1483	173	2344
Neutral	340	59	1601
Positive	162	47	3791

***Tab.** 7 Levels of the variable "Score" compared to the ones inferred by Vader (Vader variable), using -0.05 and +0.05 as cut-offs.*

Afinn Var. / ***Score Var.***	**Negative**	**Neutral**	**Positive**
Negative	2193	1142	665
Neutral	651	729	620
Positive	466	1418	2116

***Tab. 8** Levels of the variable "Score" compared to the ones inferred by Vader (Vader variable), using 33rd and 66th percentiles as cut-offs.*

Using the same method as before, we computed the "accuracy": for the couple of cut-offs [-0.05, +0.05], the accuracy is 0.53 and for the other cut-offs is 0.50: both these values are greater than the ones obtained with Afinn.

In conclusion, it seems that Vader produces better results than Afinn, especially when considering the values [-0.05, +0.05] as cut-offs to set the different labels.

Conclusions

The project explored text mining techniques, applying them to the Amazon Fine Food Reviews dataset. The researchers performed clustering, summarization, and sentiment analysis to demonstrate the potential of these methods for extracting insights from large datasets. The project successfully demonstrated the versatility of text mining for analyzing textual data to inform decision-making processes in e-commerce.

However, the researchers acknowledge the limitations of the methods when dealing with the complexity of natural language. The clustering techniques, for example, were limited in their ability to accurately group reviews based on the nuances of human expression. The sentiment analysis techniques, while effective, also encountered challenges in accurately capturing sentiment in complex cases, potentially requiring more advanced models to improve precision.

The researchers suggest future improvements could involve exploring more advanced models to address the ambiguity in language, which would require greater computational resources. Despite the limitations, the project successfully highlighted the value of text mining for analyzing large-scale textual data to improve strategic planning.