Unsupervised Clustering for Amazon Reviews

Adrian Iordache

January 17, 2021

Abstract

In this project we will try to use unsupervised methods to predict clusters for reviews on Alexa, the AI virtual assistant from Amazon. During this assignment will use comparisons with random chance and a basic supervised approach on the 'rating' feature, which is also provided in the dataset. The unsupervised methods compared for clustering are: K-Means Clustering and Gaussian Mixture Models. This project will have two main objectives: clustering reviews based on positive or negative feedback (criterion for the positive or negative feedback will be chosen based on the 'rating' feature after the dataset analysis) and clustering based on ratings from 1 to 5.

1 Introduction and Exploratory Data Analysis

The dataset consists in approximately 3000 sample of text reviews with ratings assigned from 1 to 5. A good start for us would be to check the ratings distribution as we can see below in the Figure 1.

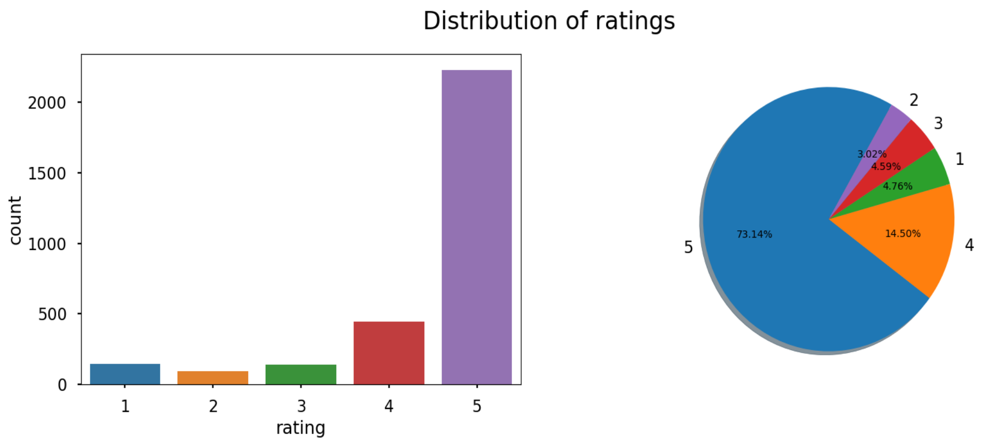

Figure 1: Distribution of ratings

In the image above we can observe one of the two main problems that will need to tackle during this assignment, namely the imbalanced labels in dataset.

The second problem is represented by data inconsistency which will be presented in section 3.

From supervised experience and short researches on the unsupervised side I reached out to the conclusion that the unsupervised methods might not be invariant when it comes to the dataset imbalance.

So the solution for this problem would be to random subsample from the dominant class, but still maintaining the limit of minimum 1000 samples from the original dataset. Another possible solution would be using only a subset of classes the clustering approaches.

Those would be the only two valid options taking into consideration the fact that supervised or unsupervised methods of subsampling would be considered cheating during this assignment.

1.1 Distribution of lengths in dataset and empty reviews

Now we can go further and analyze the distribution of number of words in reviews and the distribution of lengths of reviews over the dataset. In this stage of the analysis I observed a number of 79 empty samples. Those might be useful because can reveal information about the target variable, for example let's say that only the low or the high rating reviews are more likely to be empty samples.

But that was not the case, after checking the distribution of those samples it came out the fact that those are uniform distributed over the rating classes. This fact it's also logical because it's equally likely that you might give a 1 star rating and say nothing as giving a 5 star rating and saying nothing.

The best decision for this situation would be to remove them. In Figure 2 we can observe the distribution of review lengths after removing the empty samples.

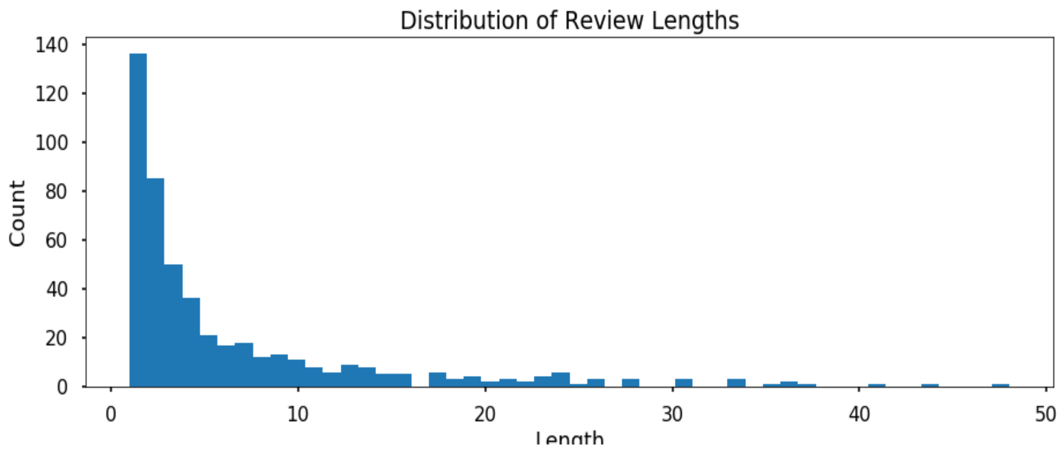

Figure 2: Distribution of review lengths in dataset

After this we can check if there are any correlation between the length of the reviews or the number of words in reviews with the target variable.

Table 1: Statistics for reviews length grouped by ratings

count	mean	std	min	25%	50%	75%	max
146.0	214.931507	213.245937	10.0	58.0	135.0	288.75	1124.0
92.0	259.978261	270.279748	3.0	95.5	167.5	317.00	1686.0
140.0	224.978571	276.581300	13.0	59.0	154.5	291.25	1954.0
447.0	181.697987	216.531226	4.0	36.0	102.0	246.00	1360.0
2246.0	110.357524	152.781962	1.0	28.0	66.0	137.00	2851.0

Table 2: Statistics for number of words in review grouped by ratings

count	mean	std	min	25%	50%	75%	max
146.0	40.746575	40.215250	2.0	11.25	26.5	57.00	230.0
92.0	50.184783	52.284288	1.0	17.75	33.0	60.25	324.0
140.0	43.650000	52.935542	2.0	12.00	30.0	57.25	361.0
447.0	35.205817	41.901204	1.0	7.00	20.0	48.50	264.0
2246.0	21.609528	29.611370	1.0	5.25	13.0	27.00	526.0

From the tables above (Table 1 and Table 2) we can see that as the rating goes higher, the review tends to become shorter.

1.2 Reviews with emojis and the distribution of those

A very common practice in Natural Language Processing is to remove the emojis and emoticons, but those might contain information about the problem we are trying to solve.

In Figure 3 we can see how the ratings are distributed in reviews that contain emojis.

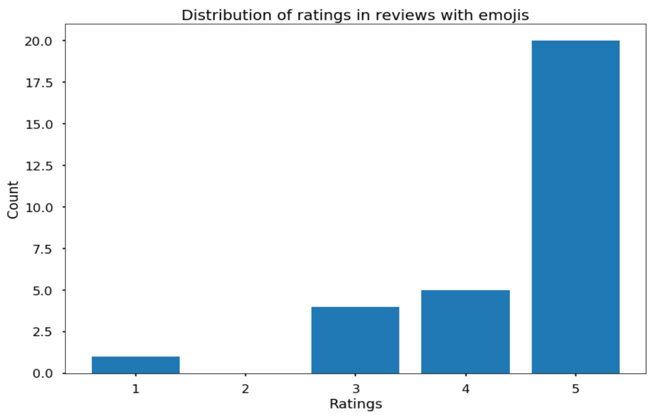

Figure 3: Distribution of ratings in reviews with emojis

Each review in the dataset contains from 0 to maximum 2 distinct types of emojis and there are exactly 30 reviews that use emojis.

As we can observe the higher rating reviews are more likely to contain emojis. And from that a better choice would be to translate those emojis in words to be later used in generating embeddings.

1.3 Word Level Ngram Analysis

This step was realized before and after cleaning text and removing frequent words with little meaning. For analysis will be using the most common unigrams, bigrams and trigrams, but in this presentation only the first two will be shown. The results can be observed in Figure 4 and Figure 5, those images are obtained after the preprocessing step which will be presented in the following section.

From the images below we can observe a separation in terms of most common unigrams and bigrams at least between the higher and the lower rating classes.

Some of the most common unigrams grouped by rating:

- Rating 1: would, buy, connect, screen, return, back, want, stop, never
- Rating 2: time, sound, would, product, speaker, buy, quality
- Rating 3: speaker, sound, quality, well, buy, great,
- Rating 4: great, love, music, sound, good, speaker, well
- Rating 5: love, great, music, easy, sound, good, buy, well, smart

Some of the most common bigrams grouped by rating:

- Rating 1: send back, ask question, customer service, waste money, never buy, great product (outlier probably)
- Rating 2: sound quality, play music, would recommend, buy another, money back, go back
- Rating 3: sound quality, could better, play music, decrease volume, request time, much time
- Rating 4: play music, sound quality, great sound, easy set, still learn, love great, easy setup
- Rating 5: easy set, prime day, great product, love love, absolutely love, love thing

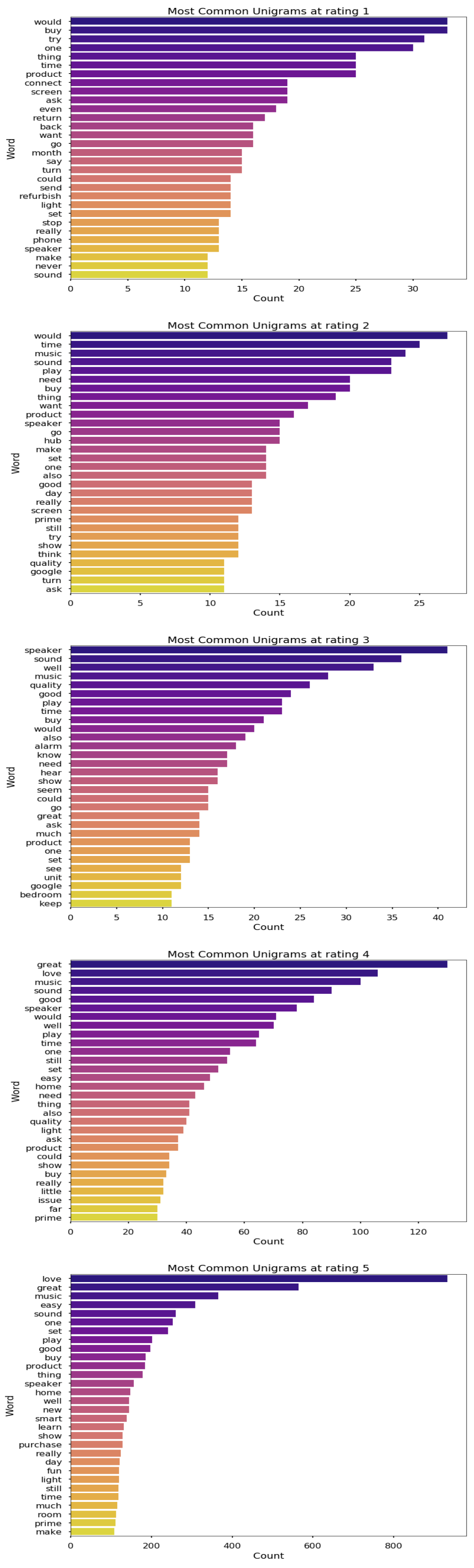

Figure 4: Word Level Unigrams

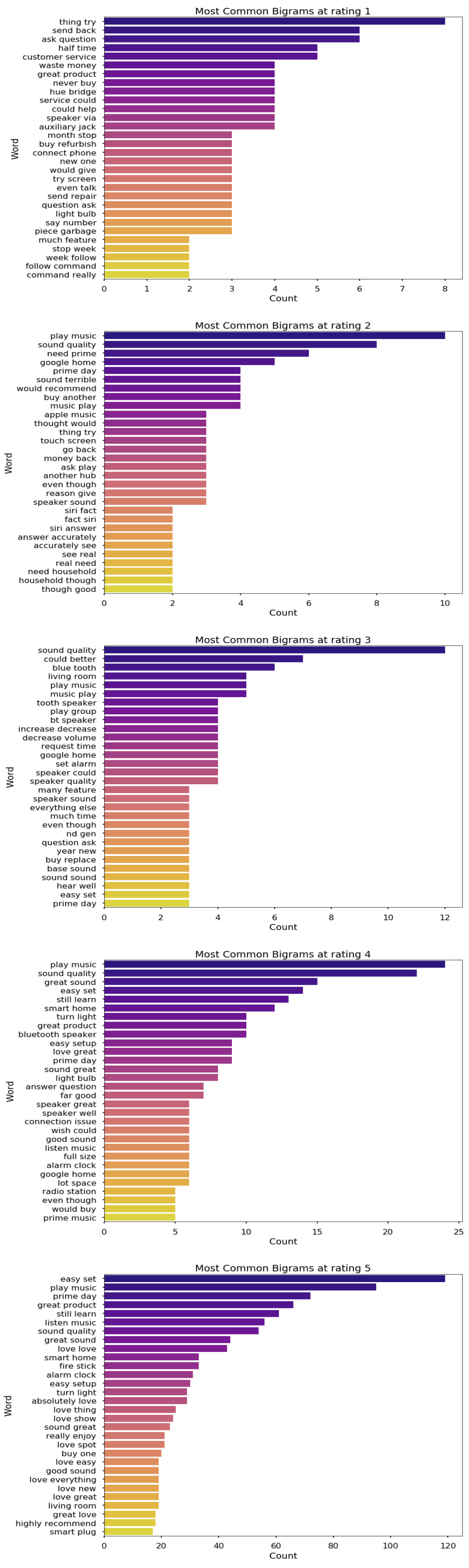

Figure 5: Word Level Bigrams

2 Text Cleaning and Embeddings

For text cleaning and preprocessing were used the following steps:

- removing URLs, HTML tags, punctuation, accented characters, white spaces
- encoding emojis to text
- expanding contractions
- tokenization
- removing stopwords
- lemmatization

After those steps for text cleaning, from previous observation on the Ngram Analysis, I decided to remove at maximum 30 from the most common words if those words are intersected in the rating categories.

For each rating category I selected the first 30 most common words, after that we take the intersection of those sets and remove them from reviews.

The reason for that is based on the following, from the Ngrams we can observe that some words with high occurrence that are uniform distributed over the rating category carry less information, words such that: 'dot', 'alexa', 'amazon', 'echo', 'device'. The final number of those type of words was 9.

2.1 Universal Sentence Encoder Embeddings

For text embeddings I decided to use Universal Sentence Encoder from tensorflow hub. The reason for that was based on the need of extracting the semantic similarity from reviews for better clustering results.

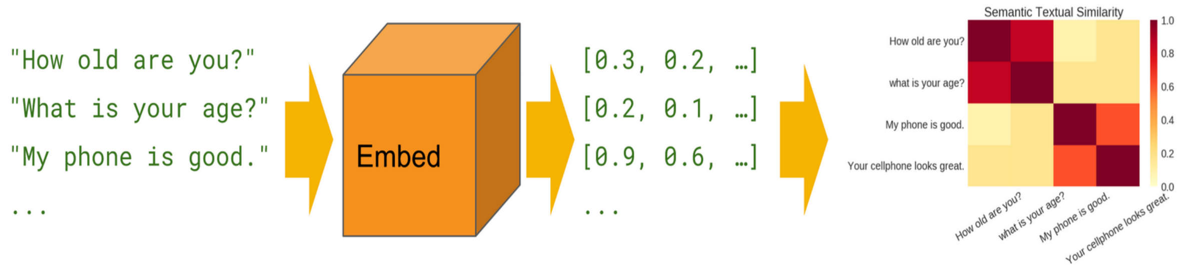

Figure 6: Semantic Similarity [1]

"The Universal Sentence Encoder encodes text into high-dimensional vectors that can be used for text classification, semantic similarity, clustering and other natural language tasks." [1].

3 Principal Component Analysis

In the end of the EDA, to get a sense of separation for each task, I used PCA and t-SNE to visualize the data points in a 3 dimensional space. In the following figures (Figure 7 and 8) will display only the PCA results on each task.

At this stage of the project we consider as feedback the classes 1, 2, 3 combined for negative feedback and 28% random samples from the rating 5 class as positive feedback, also for the clustering by rating task will use all samples in the classes 1, 2 and 3 and 25% random samples of rating 4 class and 25% random samples from rating 5 class.

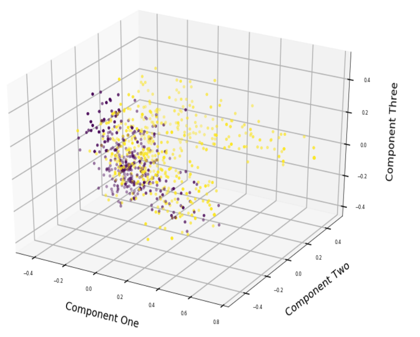

Figure 7: Principal Component Analysis on Feedback target

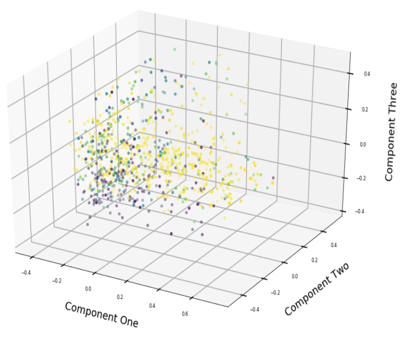

Figure 8: Principal Component Analysis on Rating target

We can observe that the problem of clustering based on feedback it's much more separable then the one for clustering rating classes.

The reason for that it's represented by the problem of data inconsistency, based on the fact that the problem we are trying to solve it's quite relative. To be more specific, there might be some separation between positive and negative samples, but for the rating target a review like 'works great' might be distributed in the rating classes 3, 4 and 5.

This type of situation where the inputs might have various outputs represents a problem in trying to find separation hyperplanes in supervised learning or clusters in data from unsupervised approaches.

4 Clustering Methods and Results

As we said in the beginning of this project for the unsupervised clustering methods will use:

1. K-Means Clustering
2. Gaussian Mixture Models

Both of those models will be compared on both tasks.

For each task, the dataset will be splitted in 3 parts sets represented by training, validation and holdout, after that will present the baseline accuracy of each unsupervised model with default parameters, followed by the same models with optimized hyper-parameters and finally a comparison with a basic supervised model represented by a standard Logistic Regression model.

It's worth mentioning that all experiments were seeded so we can be sure of reproducible results and the label mapping to clusters and optimization was done on the validation set, the following results being from a separated holdout set.

4.1 K-Means Clustering

A very informal way to thing about K-Means it's that the algorithm tries to minimize the distance within the cluster and also to maximize the distance between different clusters by an iterative process.

1. Assign centroids for each cluster (random or by some criterion)
2. Calculate a distance from all samples to the centroids
3. Assign each sample to the closest cluster
4. Find the new centroids from the mean of samples in each cluster

Steps 2, 3 and 4 are repeated until a certain threshold of tolerance it's reached or centroids are not moving.

A very important aspect of K-Means algorithm it's that even the fact that will always converge, it's not guaranteed that will find the global minimum, being very dependent on the initialization of centroids.

4.2 Gaussian Mixture Models

A Gaussian Mixture Model assumes that each sample from the dataset comes from one Gaussian distribution that describes each of the K clusters. In this approach each cluster can be represented from the following:

- A mean μ that represents the centre of the cluster.
- A covariance matrix Σ that can be interpreted as its width (or in multivariate case the dimensions of the ellipsoid)
- A probability π that defines the height of the distribution

So for each cluster k in K will have set of 3 learnable weights represented by (μ_k, Σ_k, π_k).

Let's take for example the case of three clusters.

To be more mathematical precise, we can express the density of a sample as the weighted sum of K Gaussian densities, which can be written as follows:

$$p(x \mid \theta) = \pi_1 \mathcal{N}(x \mid \mu_1, \Sigma_1) + \pi_2 \mathcal{N}(x \mid \mu_2, \Sigma_2) + \pi_3 \mathcal{N}(x \mid \mu_3, \Sigma_3)$$
$$\theta = \{\pi_1, \pi_2, \pi_3, \mu_1, \mu_2, \mu_3, \Sigma_1, \Sigma_2, \Sigma_3\}$$

And the optimization problem that needs to be solved in order to find those weights is:

$$\underset{\theta}{max} \prod_{i=1}^{N} p(x_i \mid \theta) = \prod_{i=1}^{N} (\pi_1 \mathcal{N}(x_i \mid \mu_1, \Sigma_1) + \ldots)$$
$$\text{subject to } \pi_1 + \pi_2 + \pi_3 = 1; \pi_k \geq 0; k = 1, 2, 3.$$

Or a more general form can be written as:

$$p(\mathbf{X}) = \prod_{n=1}^{N} p(\mathbf{x}_n) = \prod_{n=1}^{N} \sum_{k=1}^{K} \pi_k \mathcal{N}(\mathbf{x}_n \mid \mu_k, \Sigma_k)$$

4.3 Final Results

Table 3: Final Accuracy on Holdout set

	Random	KM	GM	KM Opt	GM Opt	LR
Feedback	50%	0.6138	0.5346	0.6138	0.7128	0.9108
Rating	20%	0.3238	0.2095	0.3428	0.3904	0.6190

- KM $\longrightarrow$ Baseline K-Means
- GM $\longrightarrow$ Baseline Gaussian Mixture
- KM Opt $\longrightarrow$ K-Means with optimized parameters
- GM Opt $\longrightarrow$ Gaussian Mixture with optimized parameters
- LR $\longrightarrow$ Baseline Logistic Regression

Best parameters for each model on feedback target:

- K-Means $\longrightarrow$ default parameters
- Gaussian Mixture $\longrightarrow$ PCA with 10 components, kmeans initialization, covariance_type diagonal,

Best parameters for each model on rating target:

- K-Means $\longrightarrow$ PCA with 10 components, n_init = 50, random initialization
- Gaussian Mixture $\longrightarrow$ PCA with 100 components, kmeans initialization, n_init = 10, covariance_type full,

5 Conclusions

As a conclusion for this assignment, we've compared two unsupervised approaches, K-Means and Gaussian Mixture, on two classification tasks based on basic Natural Language Processing knowledge trying to tackle the problem of imbalanced labels and data inconsistency.

This project resulted in a Gaussian Mixture Model better then the K-Means and Random Change on both tasks, but still not close enough to the supervised model. This represents a fair result because both of my Gaussian Mixture Models are using kmeans initialization, from that representing a better variation of the standard K-Means algorithm.

References

[1] *TensorFlow: Large-Scale Machine Learning on Heterogeneous Systems*. Software available from tensorflow.org. 2015. URL: `https://tfhub.dev/google/universal-sentence-encoder/4`.