Data Engineering

Assignment 1 – Report

Paulina Thrasher

06/03/2025

**Problem Statement**

Sentiment analysis is a technique used in Natural Language Processing to determine the tone or emotion of a piece of text. It is used frequently in business and marketing contexts to understand customer feedback on a large scale allowing key stakeholders to improve decision-making. In this example, I will analyze mock Amazon product reviews with the aim to determine whether a review can be considered positive or negative. In total there are 241 reviews utilized in this analysis. This would allow Amazon stakeholders to determine trends amongst their customer’s sentiment and ultimately improve their product offerings. The dataset consists of customer reviews, customer ratings (using a scale from 1-5) and product ID’s. The objective of this report was to create a sentiment analyzer model that allows us to categorize reviews as positive or negative and to group similar reviews together.

**Background on Sentiment Analysis**

Sentiment analysis is a common task done in text mining and involves analyzing text and automatically labeling it with an indicator of its emotional tone (e.g. positive or negative). Context can often add complexity to sentiment analysis but is a key point of consideration as it can change the way in which text is interpreted. For instance, if someone leaves a vague comment like “I like it,” the machine needs to understand what “it” refers to (Nvidia, n.d.). Additionally, things like sarcasm and irony present additional challenges as they may say one thing but mean another (Mukherjee, 2025).

For a computer to be able to analyze text, the first key step involves feature engineering, or the process of transforming text data into word vectors (Nvidia, n.d.). A basic way to represent text as a vector is to look at word counts. This technique is known as the “Bag of Words” method and represents a piece of text by simply counting how often each word appears resulting in a numerical vector where each dimension corresponds to a word. A similar method called Term Frequency-Inverse Document Frequency (TF-IDF) also looks at word frequency, however, it assigns a weight to each word based on the term frequency (how often a word appears) and the inverse document frequency (how rare a word is across documents) (Michels et al. 2025).

For this analysis, I use a transformer-based sentence embedding model from Hugging Face called t "sentence-transformers/all-MiniLM-L6-v2" to transform the product reviews into numerical vectors called embeddings. Hugging face provides different pre-trained models that make it much easier and faster to implement (Hugging Face). Embeddings allow us to capture the semantic meaning in a way that wouldn’t be possible using a traditional TF-IDF model, allowing us to compare and see similarities and differences amongst our reviews (Espejel, 2022).

**Initial Sentiment Classification**

To prepare the data, we first created a class to turn text reviews into numerical representations called embeddings so that the machine can analyze them. The ReviewAnalyzer class is designed to process text and generate these embeddings using the pre-trained model mentioned above. After embeddings are generated, the \_mean\_pooling function is used to average the embeddings of all words, creating a fixed-length vector representation for each review.

After the reviews have been transformed into numerical embeddings, these embeddings can be used to build a classifier model that uses average embeddings (centroids) to classify the sentiment of new reviews. Using the ‘rating’ column, I created binary labels in which reviews greater than 3 were labeled as positive (1) and reviews less than or equal to 3 as negative (0).

The data is split into a training and test set with 80% designated for training and 20% for testing. Next, the average embedding for positive and negative reviews is calculated. This creates two ‘centroids,’ or points in 2-dimensional space that represent the average positive review and the average negative review. The idea behind this is that new reviews will be labeled as positive or negative according to whichever centroid they are closest to. Cosine similarity is the measure used to calculate how close a new review embedding is to the positive or negative centroid. Cosine similarity is a relevant and useful measurement here, as it is computationally efficient, and it can compare similarity between two vectors regardless of their length (Michiels, 2025). To evaluate model performance, the classifier makes predictions on the test set and accuracy is calculated by comparing to the actual labels (y\_test). The results can be seen below.

The model achieved an accuracy of 77.5% meaning it classified approximately ¾ reviews correctly. The ROC-AUC score (0.774) means that the model is performing better than random guessing (0.5) but still could be improved (an ROC-AUC score of 1.00 is considered perfectly predictive). Additionally in this model there were 7 false positives (negative reviews incorrectly classified as positive), and 4 false negatives (positive reviews classified as negative).

A screenshot of a computer

AI-generated content may be incorrect.

In a second iteration of the model, embeddings are normalized using L2 normalization. This makes sure that all embedding vectors have unit length. Later in the model, when we use cosine similarity to compare embeddings, normalization allows a more accurate comparison and prevents longer embeddings from having an advantage. Seen below, the model achieved 85.7% accuracy and an ROC-AUC score of 0.86. There were also fewer misclassifications (5 false positives and 2 false negatives). The improved results suggest that the previous model trained on a single split of the data could have been skewed due to variations in the magnitudes of the embeddings.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Centroid Model with Cross Validation**

Instead of training the model on a single train-test split, I decided to next implement 5-fold cross validation using StratifiedKFold with the same centroid model. This method splits the dataset into 5 folds and uses each data point in both training and testing as it iterates from one-fold to the next. It is a beneficial technique to use as it has relatively low variance and bias (Han et al. 2012). Other than this, the rest of the process is akin to the initial centroid classifier I implemented in the previous section. Positive and negative reviews are computed from the training data, and new reviews are labeled based on cosine similarity. The major difference is that this classification is repeated across different data splits and our performance metrics (accuracy and ROC-AUC) are averaged across folds.

Results can be seen below. The model’s accuracy using 5-fold cross validation was 78.39%, a slight improvement from using a single train\_test split without normalization, suggesting that cross-validation served as a stronger training technique. The standard deviation measurement can be interpreted as a measurement of how much the model’s accuracy varied across the 5 folds. A standard deviation of 0.066 indicates slight variations across folds. A slightly higher ROC-AUC score (0.79) tells us that the classifier did a marginally better job at distinguishing between positive and negative reviews as a value closer to 1.0 indicates better performance. Finally, we can see by looking at the confusion matrix how cross-validation allows us to utilize more of our dataset in testing. In terms of incorrect labels, there were 37 false positives and 15 false negatives which shows that the model struggled a bit more with accurately designating negative reviews as negative.

A screenshot of a computer screen

AI-generated content may be incorrect.

Once again, I implemented L2 normalization on the cross-validation centroid model to ensure that cosine similarity comparisons are fair. Interestingly, the performance slightly declined, and results can be seen below. Since a cross-validation model already evaluated multiple different splits, the model was already getting a more balanced representation of the data. The minimal decrease in performance suggests that the normalization did not make a significant difference in this model.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Logistic Classifier with Cross Validation**

The last two models relied on distance-based comparisons to classifying review sentiment rather than learning from the data itself. A logistic regression classifier model is an alternative method that identifies patterns in the embeddings to make better predictions. Logistic regression is a supervised learning algorithm, meaning it utilizes a labeled dataset to classify examples (Han et. al., 2012). Unlike the previous centroid models, logistic regression learns from the data by assigning weights to different features within each review. It calculates a score based on these weights which is then transformed by a sigmoid function into a probability between 0 and 1, showing how likely it is that the review is positive or negative (Han et al., 2012).

For this model, the embeddings are standardized to have a mean of 0 and standard deviation of 1. Since logistic regression works by assigning weights, they may have different scales which can cause some features to dominate others. Standardization ensures there is a fair contribution from all features. Once again, the dataset was split using stratified K-fold cross-validation, creating a balanced distribution of positive and negative reviews in each iteration. The results showed an improvement in model accuracy and can be seen below. Mean accuracy was 90.04%, indicating that the model correctly classified 90.04% of reviews as positive or negative. The ROC-AUC score was 89.02% which is a significant improvement over the centroid models. This shows that the model is effective at differentiating between the positive and negative classes. Finally, there were 9 false positives and 15 false negatives, a lower rate of mistakes than both centroid models from the previous sections.

A screenshot of a computer screen

AI-generated content may be incorrect.

**Conclusion**

This project demonstrated how sentiment analysis can be done using transformer-based embeddings and different types of classification techniques. I started first with a basic centroid-based classifier and was able to create a simple yet understandable method for categorizing Amazon reviews as positive or negative. Through further experimentation, I incorporated cross-validation and a logistic regression classifier to improve performance.

One of the key improvements was expanding the dataset to include a greater range of reviews. With a larger dataset, models are exposed to more variation in the training data which then allows them to generalize better. For instance, the data set was initially expanded to 161 instances and the performance measurements can be seen below for centroid-based classification with a single train\_test split.

A screenshot of a graph

AI-generated content may be incorrect.

*Results from Initial Single Test-Train split model with 161 reviews*

Methods that improved these scores were normalizing the single test\_train split centroid model, implementing cross-validation and logistic regression. Cross validation showed different results because it tests the model on multiple different splits. In this case, I used 5 splits, the model is trained on 4 folds and tested on remaining 1. This process is repeated until each fold is used as the test set. As a result, different samples are used as the test set each time, which leads to slight differences in accuracy across each fold. Logistic regression was used as an improvement of this sentiment analyzer. This method learns from the data by rather than solely relying on a distance-based comparison. The logistic regression model provided the best results with approximately 90% accuracy and an ROC-AUC score of 0.89 displaying a strong ability to differentiate between positive and negative reviews.

These experiments show the importance of iteration and evaluating multiple machine learning approaches when developing a sentiment analysis model. In the future, additional improvements could be made by experimenting with other classifiers such as Support Vector Machines or Random Forest or implementing more neutral sentiment categories such as ‘neutral’ or ‘mixed’.

**Sources**

Espejel, Omar. Getting started with embeddings. (June 23, 2022). Retrieved March 5, 2025, from <https://huggingface.co/blog/getting-started-with-embeddings>

Han, J., Kamber, M., & Pei, J. (2012). Data mining: Concepts and techniques (3rd ed). Elsevier/Morgan Kaufmann.

Hugging Face. Sentence-transformers (Sentence transformers). (2024, April 30). <https://huggingface.co/sentence-transformers>

Michiels, Lien, Martens, David. (2024, November).  *Text Mining and Rule Mining* [PowerPoint slides]. Faculty of Business Economics, University of Antwerp. <https://lms.uantwerpen.be/ultra/courses/_111485_1/outline/edit/document/_5853217_1?courseId=_111485_1&view=content>

Michiels, Lien, Martens, David. (2024, November).  *Similarity, Neighbors, and Clustering* [PowerPoint slides]. Faculty of Business Economics, University of Antwerp. <https://lms.uantwerpen.be/ultra/courses/_111485_1/outline/edit/document/_5842315_1?courseId=_111485_1&view=content>

Mukherjee, Tanmoy (2024, February 25).  *Text Analysis for Business Students* [PowerPoint slides]. Faculty of Business Economics, University of Antwerp. <https://lms.uantwerpen.be/ultra/courses/_109247_1/outline/file/_6062538_1>

Nvidia. What is sentiment analysis? (n.d.). Retrieved March 5, 2025, from <https://www.nvidia.com/en-us/glossary/sentiment-analysis/>