How and Why Does AI Classify Different Emotions: A Study into Sentiment Analysis

Krish Kalla

Abstract

This research paper intends to explore the methodologies and effectiveness of artificial intelligence (AI) in classifying different emotions through sentiment analysis. Sentiment analysis, also known as opinion mining, utilizes natural language processing (NLP), computational linguistics, and text analysis to systematically identify, extract, quantify, and study affective states and subjective information. The study provides an overview of the current landscape of sentiment analysis, highlighting widely used techniques such as traditional machine-learning models, advanced deep-learning models such as Long Short-Term Memory (LSTM) and Transformer (E.g., BERT, GPT) models, as well as their shortcomings with current standards. A detailed methodology is presented, focusing on data collection, preprocessing, model training, and evaluation. The LSTM model created for this research demonstrates high performance in capturing long-term dependencies in text, marking around an 85% accuracy rate. The results underscore the significant advancements in AI-driven sentiment analysis and its applications across various industries, including marketing, customer service, and market research. Furthermore, the study highlights the potential of sentiment analysis in the mental health domain, where it can facilitate early detection and intervention for mental health issues through the analysis of textual data. The findings contribute to the ongoing development and refinement of AI techniques for more accurate and nuanced emotion classification.

Introduction

Artificial intelligence (AI) has made significant strides in various fields in recent years, with one of the most intriguing and impactful applications being sentiment analysis. Sentiment analysis, or opinion mining, involves using natural language processing (NLP), computational linguistics, and text analysis to systematically identify, extract, quantify, and study affective states and subjective information [2]. This technology has become a cornerstone in numerous industries, including marketing, customer service, finance, and social media, where understanding human emotions and opinions is crucial.

Traditional NLP approaches typically rely on predefined linguistic rules and lexicons to analyze text, which involves explicitly programming the rules that govern language processing. These methods can be quite effective for specific, well-defined tasks but often struggle with the complexities and nuances of human language, such as context, irony, and idiomatic expressions.

In contrast, the ability of AI to classify different emotions is rooted in its capacity to process vast amounts of textual data and recognize patterns that may be imperceptible to the human eye and mind. Through machine learning algorithms and sophisticated models, AI can discern the sentiment behind words, phrases, and even entire documents, categorizing them into emotional states such as happiness, sadness, anger, surprise, and more. This classification process is about identifying positive or negative sentiments and understanding the nuances and complexities of human emotions. Sentiment analysis takes a step back from this, and rather than classifying each emotion in a sentence, it only looks at a sentence's general picture, or "sentiment": positive, negative, or neutral.

However, the significance of this capability extends beyond mere classification. Understanding the 'how' and 'why' of AI's ability to classify emotions opens the door to improving these systems, making them more accurate, empathetic, and contextually aware. For instance, AI's ability to detect subtle emotional cues in text supplied by mental health agencies can provide early warnings and interventions for distressed individuals.

This study delves into the mechanisms behind sentiment analysis, exploring the algorithms and models that power these AI systems. It also examines the reasons behind their effectiveness, considering the intricate interplay of linguistic features, context, and data patterns that enable accurate emotion classification. By shedding light on these aspects, the research aims to contribute to AI's ongoing development and refinement in understanding and interpreting human emotions.

Current Featurescape

The current landscape of sentiment analysis is characterized by various techniques and tools that leverage machine learning algorithms and NLP methods. These tools are widely used across numerous industries to gain insights into customer emotions, public opinions, and social trends. There are three main ways sentiment analysis is defined in the world currently:

***Lexicon-Based Approach*:** The Lexicon-based methods rely on predefined lists of words annotated with their corresponding sentiments. These approaches are straightforward and interpretable but struggle with context and subtle nuances in language. Such approaches are used for quick scoring on social media monitoring apps or basic customer feedback analysis tools [6]. These models, however, tend to overlook contextual nuances and can give false positives/negatives when dealing with instances such as double negatives or long-term references.

***Machine Learning-Based Approach*:** Machine Learning models, such as Naive Bayes, Support Vector Machines (SVM), and Random Forests, use statistical methods to classify text based on labeled training data. These models require feature extraction techniques, such as n-grams, part-of-speech (POS) tagging, and term frequency-inverse document frequency (TF-IDF) [4]. However, these models are only feature learning, where features are given and then solved. No temporal information is stored, which reduces accuracy when trying to model sequences of text that occur in different time periods. These are significantly more versatile and accurate than the lexicon-based approach above and are mainly used in automated customer service responses, product review analyses, and market research, where temporal information isn't necessary.

Deep Learning-Based Approach**:** Recently, however, Deep Learning models, particularly those with neural networks, have gained popularity due to their ability to capture complex patterns and contextual information. Fixing the previous shortcomings with machine learning-based models, deep learning-based models store both feature and temporal data, allowing them to map out sequences of events to categorize information. Models like Long Short-Term Memory (LSTM) [1] and Transformers (e.g. BERT, GPT) are now widely used [8]. Deep learning models have largely taken over and are present in sophisticated sentiment analysis for social media platforms and nuanced emotional analysis in customer interactions.

Methodology

1) *Data Collection*

The first step was to gather a comprehensive dataset that accurately represents the various emotions we aim to classify. Such sources include:

- Social Media Platform posts from sources such as X, Facebook, or Instagram.
- Customer Reviews of products from e-commerce sites like Amazon or Yelp
- News Articles and Blogs, especially those written specifically in the opinionated sections
- Surveys and Questionnaire responses from targeted surveys in specific domains

I chose the second option, going with Customer Reviews, but rather from E-commerce sites, I took movie reviews from the popular movie review site IMDb. With over 50000 movie reviews, both positive and negative, it gave the perfect dataset to train and test a model.

2) *Pre-Processing Techniques*

Next, pre-processing the data is crucial to preparing the raw data from the dataset into something usable for the model. Each raw text must be converted into specific words, and then into tensors, or numerical representations of the text, which the model can use to understand the similarities and differences between words. The steps are as follows:

Text Cleaning**:** This process involves removing various forms of noise that can interfere with sentiment analysis. Common forms of noise include HTML tags, which are often found in web-scraped data, URLs, which do not contribute to sentiment, and special characters that can clutter the text. Effective text cleaning is essential for ensuring that the input data is clean and standardized, which is crucial for accurate sentiment classification.

Stemming and Lemmatization: These techniques reduce words to their base or root forms to standardize variations and reduce complexity. Stemming involves chopping off prefixes and suffixes to get to the root form (e.g., "running" becomes "run"), while lemmatization considers the context and converts words to their base form (e.g., "better" becomes "good"). This standardization helps the model recognize different forms of the same word as a single entity, improving its ability to learn and make accurate predictions.

Stop Word Removal**:** This step involves eliminating common words that do not carry significant meaning or contribute to the sentiment of the text. Examples of stop words include "and," "the," "is," and "at." These words are filtered out because they appear frequently and provide little to no useful information about the sentiment. Removing stop words helps reduce the dimensionality of the data and focuses the model on more relevant features.

Punctuation and Negations**:** Proper handling of punctuation and negations is crucial for maintaining the context and meaning of sentences. For instance, the presence of a negation word like "not" can completely change the sentiment of a phrase (e.g., "happy" vs. "not happy"). Ensuring that these elements are correctly interpreted helps the model understand the true sentiment being expressed. Special attention is needed to ensure that punctuation marks are not misinterpreted or ignored, as they can also alter the meaning of a sentence.

Tokenization**:** This process involves splitting the text into individual words, known as tokens. In my project, I used full-word tokenization, where each word in a sentence is treated as a separate token. This method is straightforward and efficient, making it suitable for my project’s scope. However, other types of tokenization, such as sub-word tokenization, can capture more detailed semantic nuances by splitting words into smaller units (e.g., "unhappiness" into "un," "happi," and "ness"). While sub-word tokenization can improve model performance, it is also considerably more computationally intensive, less efficient to implement, and requires a much larger dataset to show its full performance, which is why I opted for full-word tokenization.

Special Tokens**:** In addition to the regular tokens, special tokens are used in the preprocessing stage to handle specific requirements of the model. Padding tokens are particularly important when dealing with sentences of varying lengths. By adding padding tokens to shorter sentences, we ensure that all sentences in a batch have the same length, allowing for efficient batch processing. Other special tokens might include start and end tokens to denote the beginning and end of a sentence, or unknown tokens to represent words that are not found in the vocabulary. These special tokens help maintain the structural integrity of the data, facilitating more effective learning by the model.

3) *Model Selection*
Third, a model is selected, and for the purpose of this research, I created an LSTM from scratch, which is a type of recurrent neural network that can capture long-term dependencies in text. [1] The LSTM is particularly well-suited for sentiment analysis because of its ability to remember important information over long sequences of text while forgetting irrelevant data, addressing the vanishing gradient problem commonly encountered in traditional RNNs. The model is composed of the following layers:

Embedding Layer**:** Converts input text into dense vector representations. Word embeddings like Word2Vec [3] or GloVe [5] are commonly utilized to capture semantic meanings. In my case, I created and trained my own embedding layer, transforming each word into a vector that assigned vectors to words based on their similarities and differences, with similar words having similar vectors, and vice-versa.

LSTM Layers**:** Multiple LSTM layers are stacked to enhance the model's ability to learn complex patterns in the text. Each LSTM layer consists of a series of LSTM units, each capable of maintaining an updatable memory state.

Dropout Layers**:** Dropout is applied between LSTM layers to prevent overfitting by randomly setting a fraction of the input units to zero during training.

Dense Laye**r:** A dense layer is added after the LSTM layers to aggregate the learned features and perform the final sentiment classification. The output layer uses a softmax activation function for multi-class sentiment classification.

4) *Training Process*
Training the LSTM model requires the pre-processed text from before to be fed into the model along with its label, in this case, whether it is a positive or negative sentence. From this, the model's hyper-parameters are chosen, which in my case were the number of LSTM units, batch size, number of epochs (or full pass-throughs of the code), learning rate, and dropout rate, and the model's weights are tuned using stochastic-gradient search methods to optimize during the training process. Additionally, an optimizer is chosen, in this case, the Adam optimizer, known for its efficiency and adaptive learning rate properties to minimize categorical cross-entropy loss, a widely-used mathematical metric used in machine learning to calculate how accurately the model performs over time. This also heavily depends on the dataset, as imbalanced datasets, with significantly more positive/negative instances, would skew the results from the expression, thus rendering its output incorrect. Finally, an output of 'positive' or 'negative' is returned and checked with the given label. Correct outputs are used to show the model what it is doing right, and incorrect outputs are used to show the model what should be changed. Based on these

changes, the model continues to try and correct itself until reaching the end epoch, where the model's state is saved, and weights are no longer tuned.

5) *Evaluation Metrics*
Once the model has been trained, a test set is then used to assess its performance. The evaluation metrics include accuracy, loss, precision, recall, and confusion, which provide a comprehensive understanding of the model's classification capabilities. The results are compared with baseline models, such as traditional machine learning classifiers and simpler neural networks, to demonstrate the effectiveness of the LSTM model in capturing long-term dependencies and improving sentiment classification accuracy.

Results

1) Performance of the LSTM Model

The LSTM's performance was evaluated using its accuracy and loss, calculated for both the training and validation datasets. The model was trained over 15 epochs on an NVIDIA 680.

Accuracy: The LSTM model achieved an overall accuracy of around 84% with a training accuracy of 87.4% and a validation accuracy of 83.9%. This was to be expected due to the relatively limited amount of training data I had compared to much larger LSTMs and the sources of error, which will be discussed further in the paper.

Loss: The LSTM model achieved a loss of 0.312 during training and a loss of 0.362 during validation

Although these numbers are quite low compared to many LSTMs, it shows that a relatively good result can be achieved using minimal hardware, time, and data, though data amounts higher than this could significantly boost the model's performance. As shown in Figure 1, it can also be seen that the model had a tough time reducing its training loss for the first 8 epochs, but after, it had a stark decrease in training loss before the epoch count ended. With more time and data, it can be surmised that it could reach losses lower than what is shown.

2) *Error Analysis*

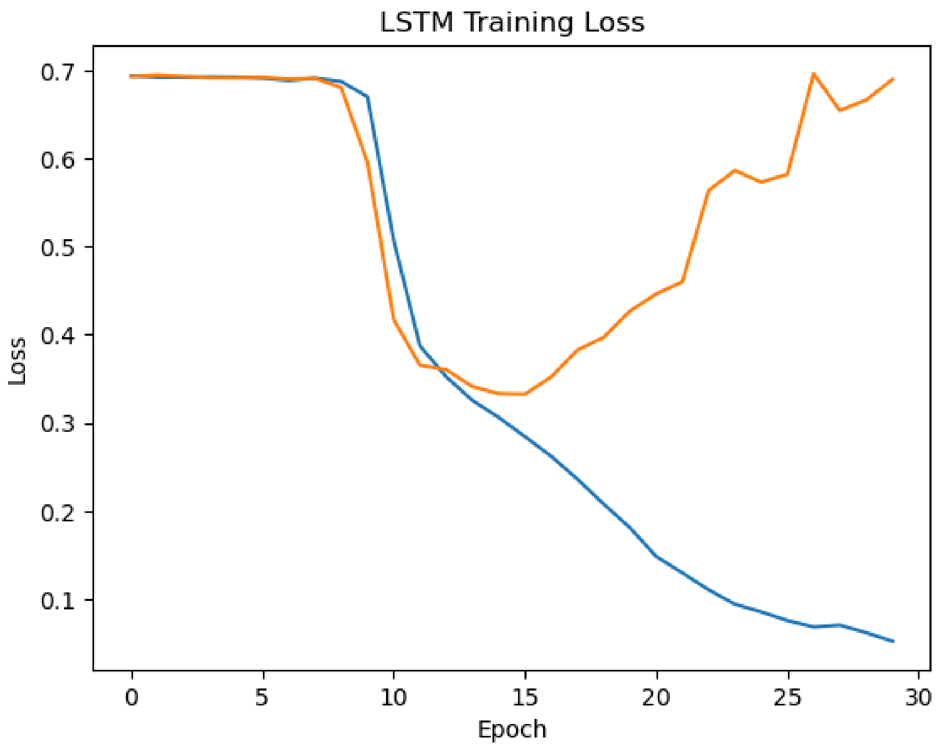

Figure 1: Blue is Training Loss, Orange is Validation Loss

As shown in Figure 1, around Epoch 15 was where the best Training and Validation Loss occurred, with the blue line showing Training Loss and the orange line showing Validation Loss, which was thereby chosen as the number of epochs the final model was trained for. Although the training loss continued to decrease, and its accuracy reached a staggering 97.8%, the validation accuracy started to decrease, reducing from the average of 87% at epoch 15 to around 83% by epoch 30. This phenomenon is known as overfitting, where the model's real-time performance starts to decrease because the model's weights fit too closely to the training data, thus only looking for similarities in the sentences’ patterns and structure rather than the semantics.

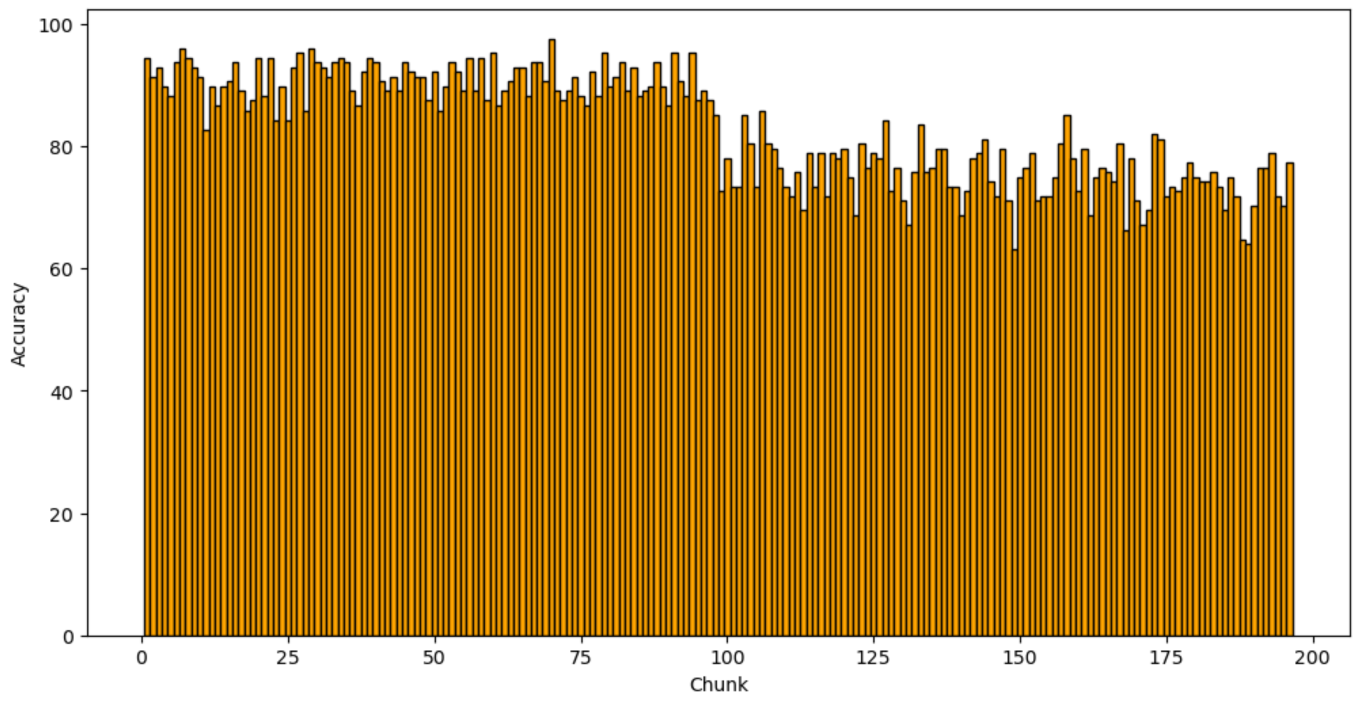

Figure 2: Batch Accuracies: Batches 1-98 is Positive, Batches 99-196 are Negative

Looking at the accuracies by batch, we can see that the batches mainly dealing with positive classification had a much higher accuracy than the batches with a mainly negative classification. This implies that the model had a more difficult time classifying negative sentences than positive, which could have been caused by the following:

Sarcasm: The model struggled with correctly interpreting sarcastic remarks and sometimes misclassified them as positive, and as many movie responses are written sarcastically, this did have a significant impact on the model's ultimate accuracy. Sentences such as "This movie was really great if you were a sloth." are quite easy for us humans to understand as being sarcastic and negative; for an AI, there are no definite clues for it to pick up on this negative connotation, probing a positive misclassification.

Multilingual Text: Sentiment analysis for texts containing multiple languages can pose difficulties, as the model's word embeddings were primarily trained in English, though this specific case did not occur as the model's dataset was primarily in English, and any multi-lingual sentences had words, such as 'primo,' the Spanish word for cousin, which did not affect the actual sentiment. This situation would also affect both positive and negative sentences equally unless a sharp increase in multilingual text was present in the negative set.

Contextual Nuances: Subtle nuances in longer texts sometimes led the model to incorrect sentiment predictions, such as previous negatives negating a positive further down. Double negatives such as the phrase "can't not," can throw off the model's prediction. Contemporary references such as "downing a bottle" can prove difficult for the model to define as negative, as these are mainly things humans have assigned connotations to over the years.

These problems are especially noticeable when viewed from a confusion matrix, as shown in Figure 3, as the percentage of negative labels being misclassified is significantly higher than with positive labels, further grounding the suspicion of sarcasm and contextual nuances as being the main players of misclassification.

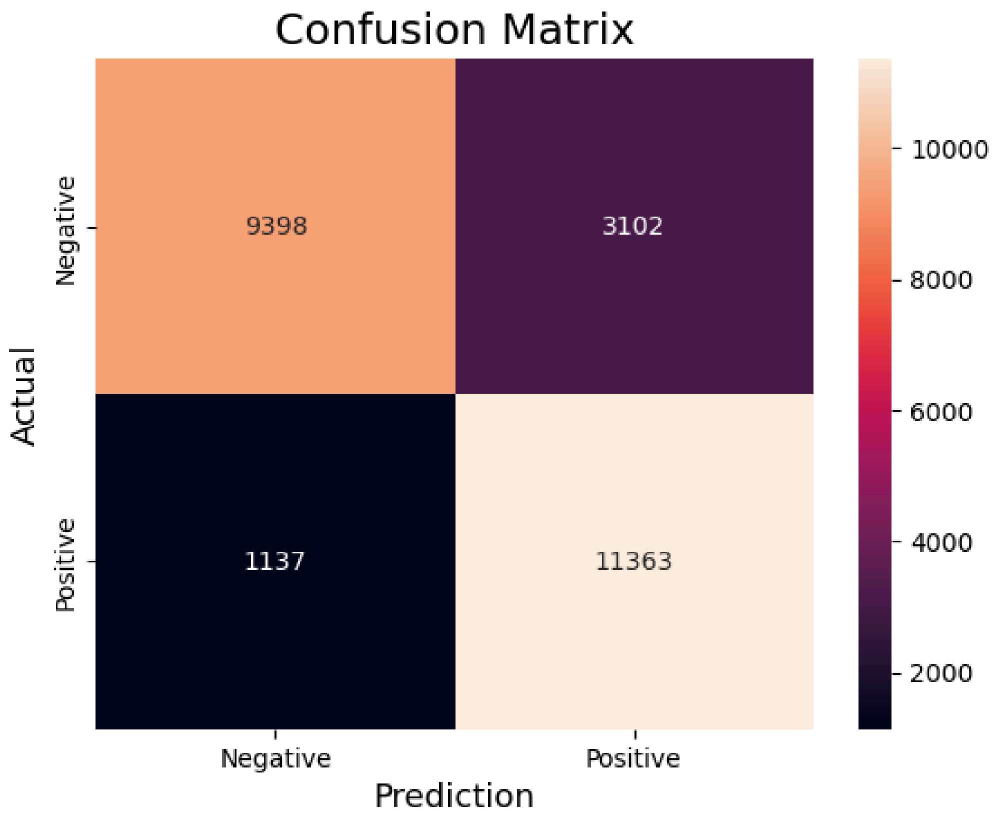

Figure 3: Confusion Matrix showing the relation between the predicted label and the actual label

Discussion

With this LSTM model, I took the set of data given with the IMDb dataset and tried mapping how each of these sentences are related to each other. With this, I used a t-distributed stochastic neighbor embedding method, or TSNE for short. [7] TSNE takes in the embedding matrices of each of these sentences present when the model was being trained and tested, and maps them on a 2d space to group similar sets of data together. This procedure produced Figure 4, which seemed odd at the time, as the positive and negative sentences didn't seem to be separated by any distinct position on the map, rather mixed together, especially nearing the center of the map. It was also interesting to see that the two datasets were split into two clusters, but not because of their sentiment. To further understand this, points in close proximity to each other were clustered together, splitting the graph into two distinct clusters, as shown in Figure 5.

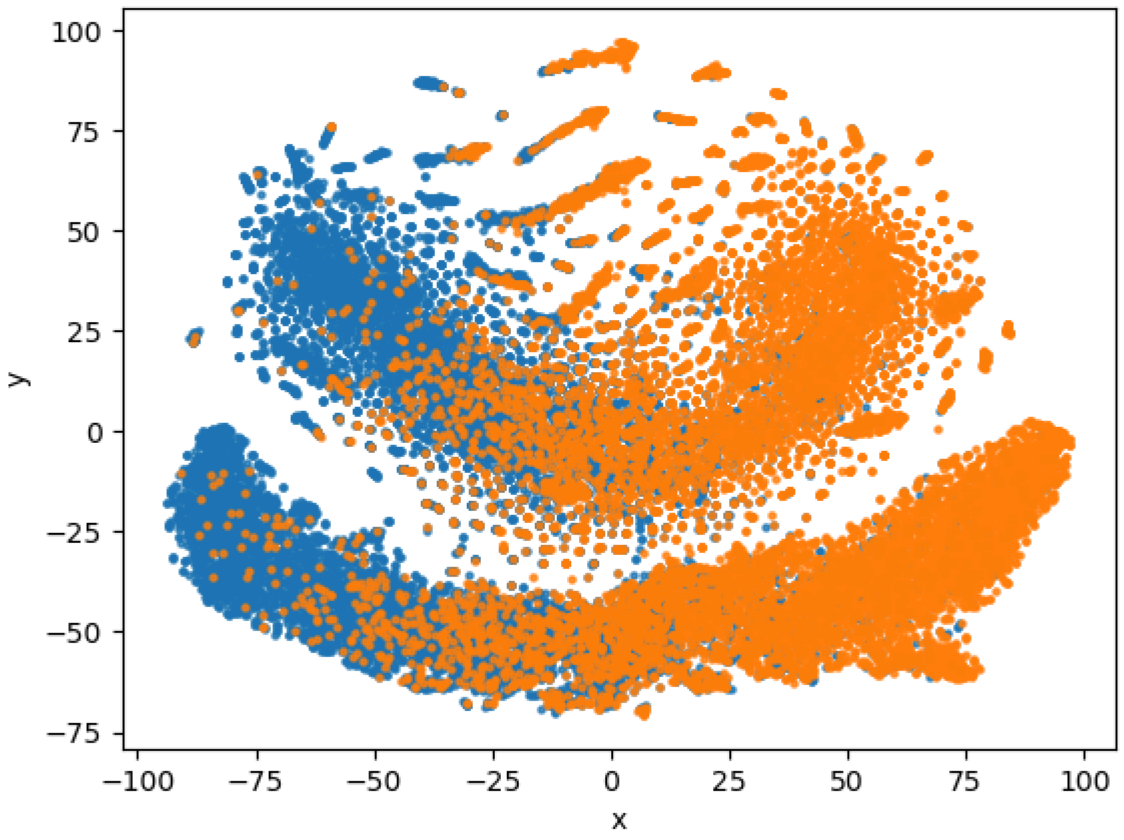

Figure 4: Sentiment Graph - Blue is Positive, Orange is Negative

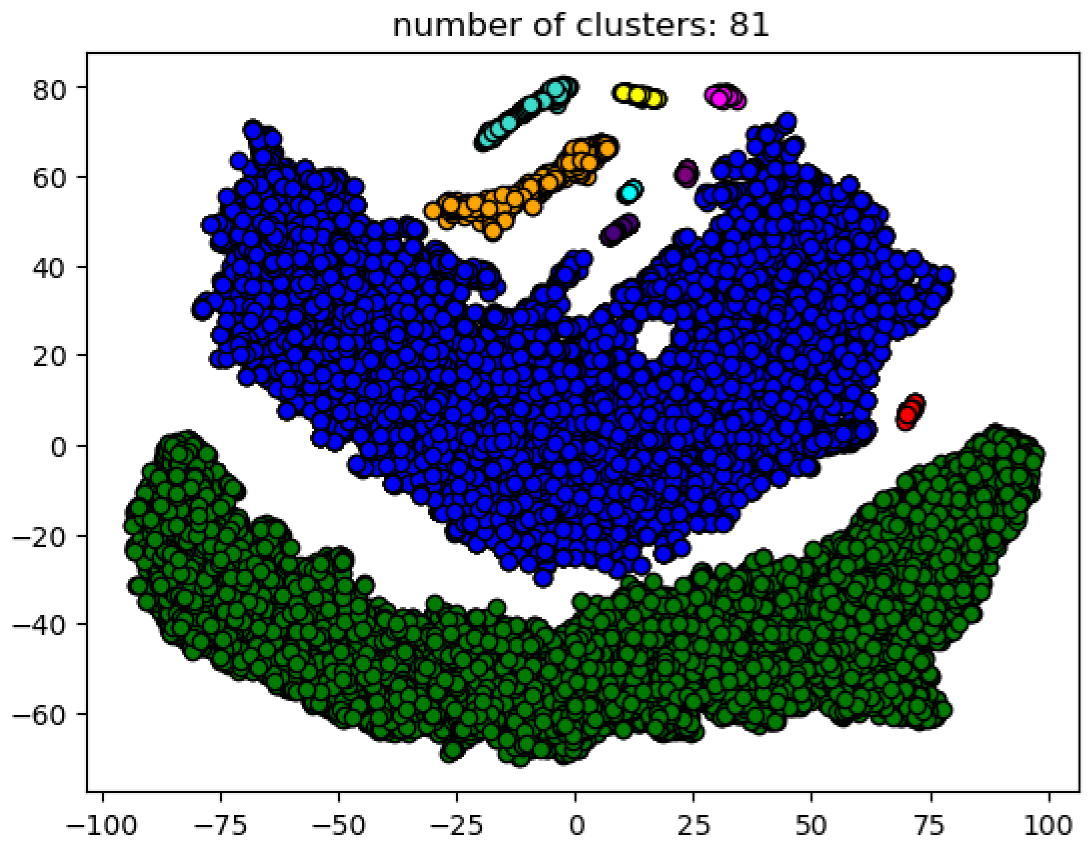

Figure 5: Cluster Graph - Blue is 'Cluster 1' and Green is 'Cluster 2'

Next, to understand why these clusters were appearing, I created Word clouds for each of them to model what topics and words were most frequently used within the clusters, producing Figures 6 and 7. These Word clouds show the most frequent words within each of the clusters, so, surmising that they should provide some detail into why they had grouped the clusters apart, I tried feeding through the common words within but could not find any defining factors that set the two clusters apart. I then created word charts based on the initial sentiment classifications rather than the clusters, producing Figures 8 and 9, and both were reasonably different, as expected, with the positive word cloud showing words like good and great prominently and the negative word cloud showing words like bad, but also positive adjectives like good, which was coupled with negating words ('not,' etc.) in many of the sentences, and also used sarcastically quite frequently.

These made me conclude that the embedding of each sentence, although directly correlating to its sentiment, cannot accurately distinguish the two sentiments based on the clustering between different sentences, and only its relative position can be considered. This does seem counter-intuitive at first, but going back to the error analysis, sarcasm was a large thing at play, and most sarcastic sentences have "positive" defining features within them, but we, as humans, have learned to understand when something is sarcastic, and put a negative connotation to that sentence. Secondly, in Figures 6 and 7, both word charts seemed to have very similar sets of words, with the largest words in both being "film," "movie," "well," and "like." "Well" and "like" are both predominantly positive adjectives, but coupled together with negating words, such as "not," it can change the meaning of a sentence drastically and turn an otherwise positive sentence into a negative one. However, because only one word might have been different, it will still be very similar to something of positive sentiment, so the two might be placed closer than two positive sentences of different subjects.

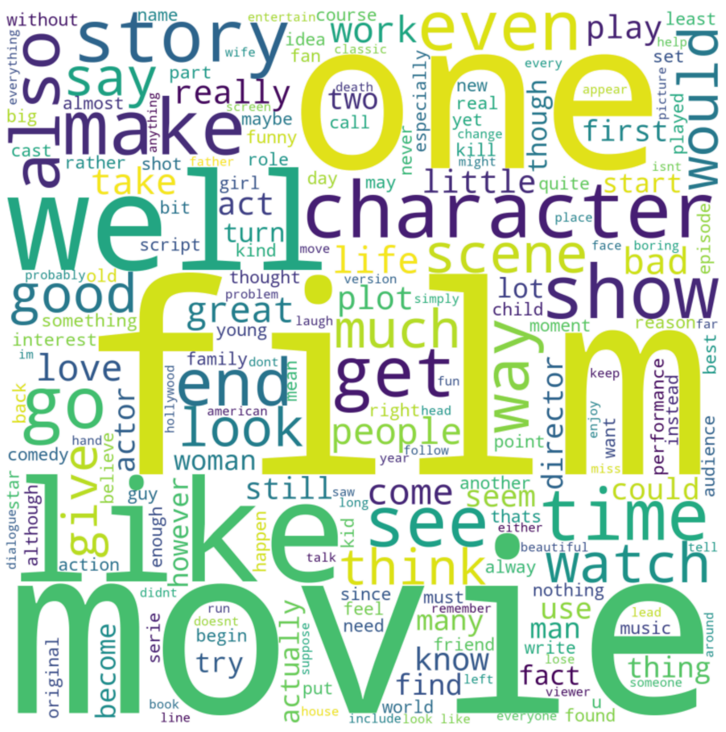

Figure 6: Word Cloud for Cluster 1

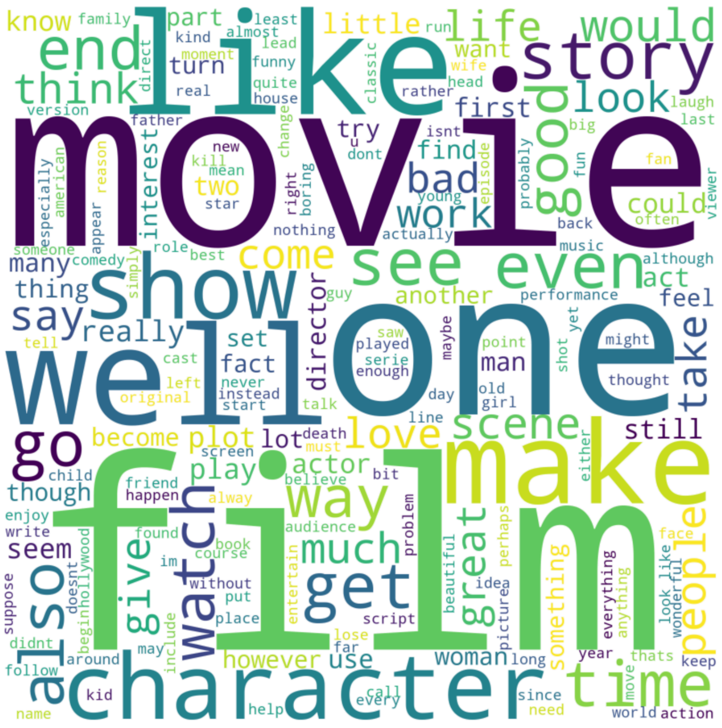

Figure 7: Word Cloud for Cluster 2

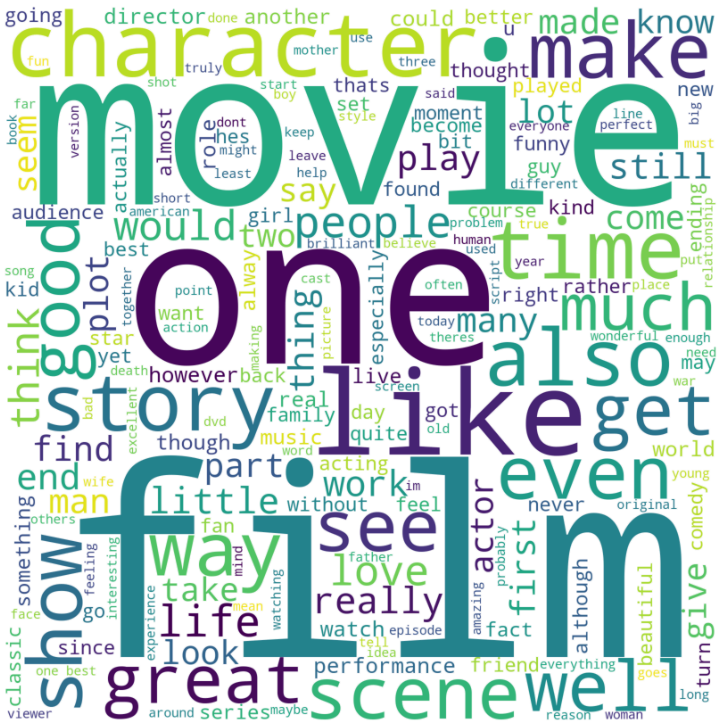

Figure 8: Word Cloud for Positive Words

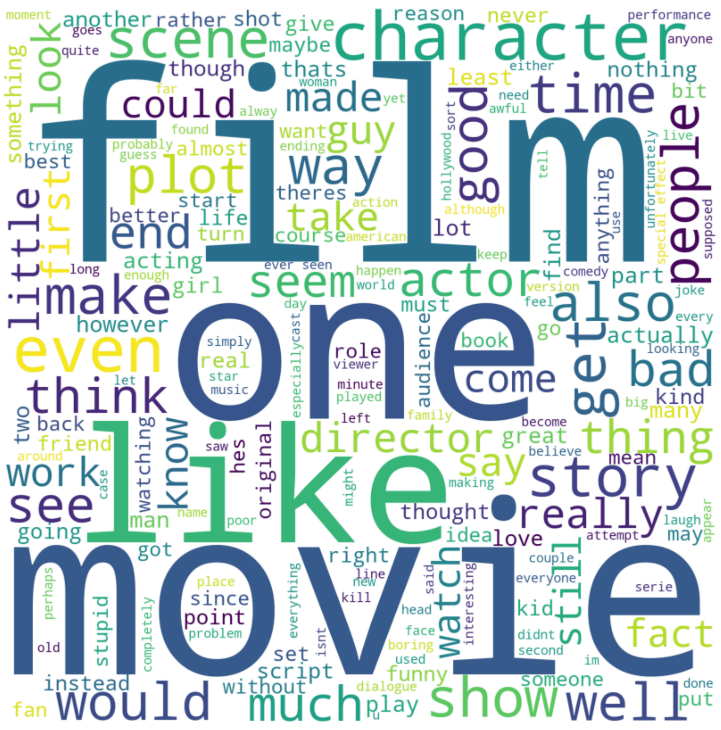

Figure 9: Word Cloud for Negative Words

Conclusion

As stated in the discussion, the model's main training and testing item, the sentence embedding, cannot directly correlate to whether it is positive or negative based on its grouping relative to other sentences. This is because the embeddings primarily capture semantic information rather than explicit sentiment orientation. Consequently, without additional context or sentiment-specific features, the model will struggle to accurately discern the emotional tone of a sentence. This is why models are trained with a sentiment given, so that each embedding can be "mapped" through similarity positional, but specific clusters, as shown with the DBScan, cannot be taken into account, and only whether it is on the "left or right side" of the graph (Figure 4) will really discern a sentence's sentiment.

Sentiment analysis, however, doesn't need to be limited to just Positive, Negative, or Neutral. As it functions based on the similarity of sentences, these connotations could be substituted with any number of emotions, such as "Happy," "Sad," "Angry," etc. So long as the dataset has a plethora of sentences to discern the subtle differences between each emotion, the model will be able to classify each of the sentences to a relatively high accuracy, most likely near or above my model. This flexibility allows for more nuanced emotional analysis, which can be particularly valuable in applications such as mental health monitoring, where understanding a wide range of emotions can provide deeper insights into a patient's emotional state. By analyzing the text data from social media posts, chat logs, and patient interactions, sentiment analysis can prove an effective early detection of mental health issues, such as depression, anxiety, and stress. This proactive approach enables timely intervention and support, ultimately improving mental health outcomes. As AI continues to evolve, its capabilities to understand and interpret human emotions will become even more sophisticated, paving the way for more empathetic and responsive technologies.

This paper underscored the ability and significance of sentiment analysis, leveraging LSTMs to allow a more nuanced understanding of one's emotional standpoints. As AI continues to evolve, ongoing research will be crucial in refining these models, ensuring they become more effective at understanding and interpreting the complexity of human dialogue and fixing error-propagating instances, such as sarcasm, to make sentiment analysis more accurate and useful in the coming years.

References

1. Hochreiter, S., & Schmidhuber, J. (1997). Long short-term memory. Neural Computation, 9(8), 1735-1780. https://www.mitpressjournals.org/doi/abs/10.1162/neco.1997.9.8.1735 (Accessed: 2024-06-02)
2. Liu, B. (2012). Sentiment analysis and opinion mining. Morgan & Claypool Publishers.
3. Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781. https://arxiv.org/abs/1301.3781
4. Pang, B., & Lee, L. (2008). Opinion mining and sentiment analysis. Foundations and Trends in Information Retrieval, 2(1-2), 1-135. https://www.nowpublishers.com/article/Details/INR-011 (Accessed: 2024-06-02)
5. Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. In Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP) (pp. 1532-1543). https://www.aclweb.org/anthology/D14-1162/
6. Taboada, M., Brooke, J., Tofiloski, M., Voll, K., & Stede, M. (2011). Lexicon-based methods for sentiment analysis. Computational Linguistics, 37(2), 267-307. https://www.mitpressjournals.org/doi/abs/10.1162/COLI_a_00049 (Accessed: 2024-06-02)
7. van der Maaten, L., & Hinton, G. (2008). Visualizing data using t-SNE. Journal of Machine Learning Research, 9, 2579-2605. http://www.jmlr.org/papers/volume9/vandermaaten08a/vandermaaten08a.pdf (Accessed: 2024-06-02)
8. Young, T., Hazarika, D., Poria, S., & Cambria, E. (2018). Recent trends in deep learning based natural language processing. IEEE Computational Intelligence Magazine, 13(3), 55-75. https://ieeexplore.ieee.org/document/8416973 (Accessed: 2024-06-02)