Movie Sentiment Analysis Final Project

Hasan Khwaja and Rishub Talreja

Github Link: https://github.com/hasankhwaja/Movie-Sentiment-Analysis-

Abstract:

This project explored the sentiment analysis using natural language processing models to classify the sentiment of randomized movie review phrases on a scale of 0 to 4 in which 0 I the most negative and 4 is the most positive. This project explored 3 models, naïve bayes classifier, support vector machines and BERT. These models were selected for their efficiency and capability to understand text sentiment. Each model went through preprocessing, training and testing. This resulted in BERT outperforming all the other models with 70% accuracy where naïve bayes classifier and SVM followed with 55% and 59% accuracy respectively. This project was conducted with the goal of understanding the emotion behind human language while serving as a clear foundation for natural language processing.

Overview:

This project aimed to create natural language processing models that would be able to take in phrases from scrambled movie reviews and perform a sentiment analysis. The sentiment analysis performed, by each model, on each phrase with the goal of producing a given sentiment score from 0 to 4 in which 0 was the most negative and 4 was the most positive. The task for the project was chosen due to how it was able to provide a solid foundation to work in natural language processing models while having a clear and concrete goal to work towards.

This problem was found to be interesting due to how natural language processing is widely utilized in machine learning given its ability to understand human language. When looking at sentiment analysis specifically, it defined a concrete application of natural language processing in which machine learning models could be used to understand the emotion behind human language. With the exception of movie reviews, use cases for sentiment analysis may include an analysis of social media reputation, customer service issues, and political analysis through media.

To tackle the problem, three NLP models were utilized. The models used were naïve bayes classifier, support vector machine and BERT. These models were chosen due to how they were responsible for efficient implementations while also being able to capture sentiment in text effectively.

Naïve bayes classifier was used due to how it was easy to implement and computationally efficient, which was especially needed given the large size of the dataset. SVM was used due to how it is known for its strong performance in high-dimensional text data while also maintaining some form of simplicity and assuming feature independence in order to gain an understanding of the text. BERT was used in order to leverage a pre-trained model that would be able to understand context more effectively in order to achieve as much accuracy as possible

Experiment setup:

The dataset consists of 156,060 entries with four columns: PhraseId, SentenceId, Phrase, and Sentiment. Each row represents a phrase extracted from a movie review, along with its corresponding sentiment label. Sentiment is categorized into five classes: 0 (negative), 1 (somewhat negative), 2 (neutral), 3 (somewhat positive), and 4 (positive). All columns had complete data with no missing values. The class distribution is as follows

- Class Distribution:
    - Neutral (2): 51.0%
    - Somewhat Positive (3): 21.1%
    - Somewhat Negative (1): 17.5%
    - Positive (4): 5.9%
    - Negative (0): 4.5%
- Average Phrase Length: Each phrase has an average length of approximately 40 characters.

As seen, the dataset was imbalanced, with neutral phrases making up the majority, and extreme sentiments (positive and negative) being underrepresented. To combat this issue, we used "*from imblearn.over_sampling import RandomOverSampler ", as shown in balanceDataset.py.* The analysis on the balanced dataset is that it consists of 397,910 entries. The class distribution is as follows:

- Neutral (2): 20.0%
- Somewhat Positive (3): 20.0%
- Somewhat Negative (1): 20.0%
- Positive (4): 20.0%
- Negative (0): 20.0%
- Average Phrase Length: Each phrase has an average length of approximately 51 characters.

For the SVM model, the preprocessing and model parameter choices in the SVM implementation are designed to optimize performance for sentiment analysis. First, the text is converted to lowercase to ensure case-insensitivity, allowing words like "Great" and "great" to be treated as the same. Tokenization is then used to break each phrase into individual words, which are lemmatized to reduce them to their base forms (e.g., "running" becomes "run"). Stopwords such as "and" or "the" are removed, as they do not contribute meaningful information to sentiment analysis. These preprocessing steps ensure that only significant words are retained, reducing noise and focusing on the features.

For feature extraction, TF-IDF vectorization is applied, transforming the text into numerical features by emphasizing words that are important to individual documents while down-weighting those common across all documents. Limiting the vocabulary to the top 5,000 features helps balance computational efficiency while retaining the most important terms.

The SVM model uses a linear kernel, which is well-suited for text data, as high-dimensional text data is often linearly separable. The regularization parameter (C) is set to 1.0, a standard value that strikes a balance between maximizing the margin and minimizing classification errors, ensuring the model generalizes well without overfitting. The gamma parameter is set to 'auto' because it is not used by the linear kernel, but this setting ensures compatibility if a different kernel is chosen in the future. Finally, the degree is set to 3, the default for polynomial kernels, although it is not actively used in this case due to the linear kernel. These choices reflect common best practices for text classification, aimed at building a reliable and efficient sentiment analysis model.

For the Naive Bayes model, the same preprocessing steps were made as in SVM. For feature extraction, CountVectorizer is used with a limit of 3000 features, which helps balance model complexity and computational efficiency. The vocabulary created by the vectorizer is used to track word frequencies, and the counts are stored in a dictionary for each sentiment label. The Laplace smoothing technique is applied to handle words that might not appear in both the training and test datasets, ensuring that the model does not assign zero probability to unseen words. This smoothing helps prevent overfitting by adjusting probabilities when encountering unknown terms.

In the BERT model the text data is first preprocessed by cleaning HTML tags and unwanted characters using the text_cleaning function, ensuring the text is standardized for model input. Missing values in the "Phrase" column are handled by replacing them with empty strings to avoid any issues during training.

The BERT tokenizer is used to convert the text into tokens, which are the basic units that the model processes. The sentences are tokenized and encoded into TensorFlow datasets, with padding, truncation, and sequence length adjustments done to fit the model's input requirements. The encoded data is then converted into batches for efficient processing during training.

The BERT model is initialized with the pre-trained bert-base-uncased model, designed specifically for sequence classification tasks like sentiment analysis. The model is set up with five sentiment labels (from "Very Negative" to "Very Positive"), and an exponential decay learning rate schedule is used to gradually reduce the learning rate during training, helping the model converge effectively. The Adam optimizer, along with sparse categorical cross-entropy loss, is employed to optimize the model's performance, with early stopping implemented to prevent overfitting. The model is then trained on the training dataset for three epochs and evaluated on the validation set.

Compared to simpler models like Naive Bayes, BERT's ability to understand complex language relationships through deep contextual embeddings allows it to outperform such models in accuracy. Although BERT requires more computational resources and time to train, it provides better results for tasks involving nuanced understanding of language, making it a powerful tool for sentiment analysis.

Experiment results:

Using Navie Bayes, we got a prediction rating of 56% accuracy. SVM got 59% accuracy and B.E.R.T. got 70% when no fine turning was being used. After some fine tuning was applied, we got 76%.

The results obtained from the sentiment analysis task using the BERT model have shown promising performance, with the model achieving a reasonable level of accuracy on the validation set. However, while the model's predictions are quite strong in terms of correctly classifying the sentiment of movie reviews, there are still potential areas for improvement. The model may be prone to errors in cases of ambiguous or nuanced language, where simpler models, such as Naive Bayes, might struggle less in comparison, though at the cost of overall performance and accuracy.

However, the results could have been higher for a few reasons. One reason would be BERT models require substantial computational resources, and the relatively limited number of epochs (three) might not have been enough for the model to fully converge to the optimal solution. The model was not able to run using GPU, as we could not get the model to run on the GPU using the discovery cluster. We had asked for help from the TA and IT. IT said they would be able to help us after the project was due. Thus, this significantly limited our ability to try model as we were under resource constraints and time limitations.

Given the choice of BERT for this task, the accuracy we achieved is generally expected, especially considering the constraints of available time and computational resources. If we had the opportunity to utilize a large language model (LLM), such as GPT-4 or similar advanced models, we would likely see significant improvements in the results. These models would likely have handled the nuances in sentiment classification even better, providing a much more refined approach to this task. Unfortunately, due to time limitations and resource constraints, we were unable to employ such models in this case.

Discussion:

To improve the results, additional steps could include fine-tuning the model further, increasing the number of training epochs, and employing more advanced data augmentation techniques. Further, experimenting with more complex model architectures, such as those incorporating large-scale transformers or incorporating domain-specific pre-training, could yield better performance. Lastly, the use of GPU resources would significantly reduce the training time, allowing for experimentation with larger models or fine-tuning strategies to achieve better accuracy.

Conclusion:

This project demonstrated the effectiveness of natural language processing models for sentiment analysis on a dataset of movie review phrases, highlighting their varying strengths and limitations. Among the models, BERT achieved the highest accuracy of 76%

after fine-tuning, outperforming Naïve Bayes and SVM, which achieved accuracies of 56% and 59%, respectively. BERT was shown to be the most suited model for understanding and performing sentiment analysis given how computational constraints, including the inability to utilize GPU resources, limiting the number of training epochs and fine-tuning opportunities, likely affecting the model's optimal performance. Despite these constraints, this project was able to effectively utilize NLP models for sentiment analysis.