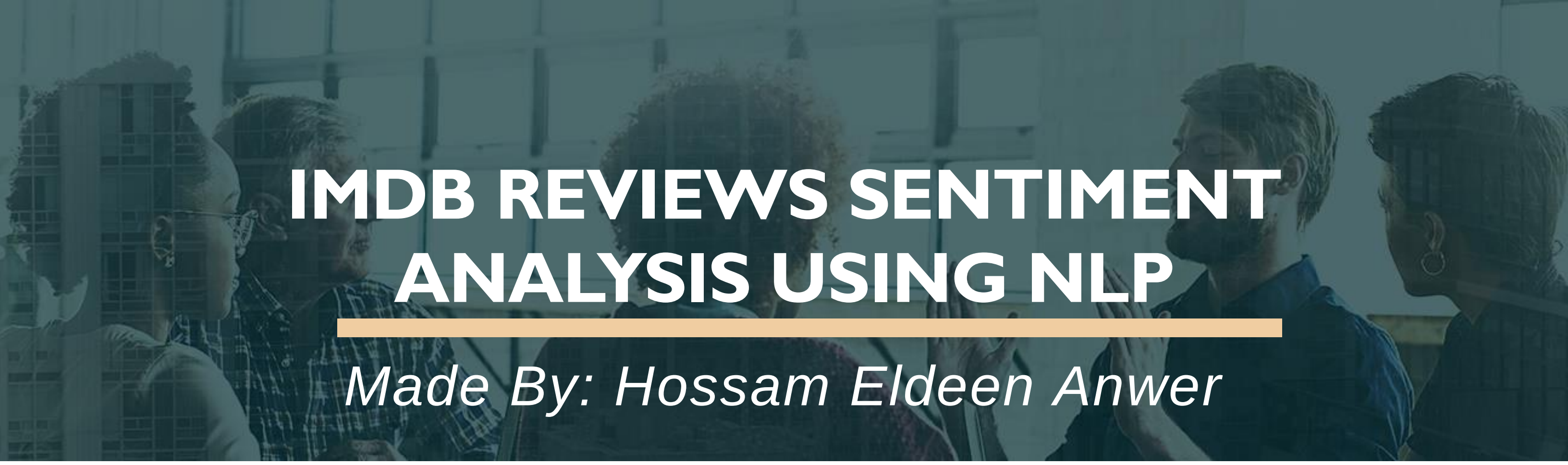


IMDB REVIEWS SENTIMENT ANALYSIS USING NLP

Made By: Hossam Eldeen Anwer

Introduction

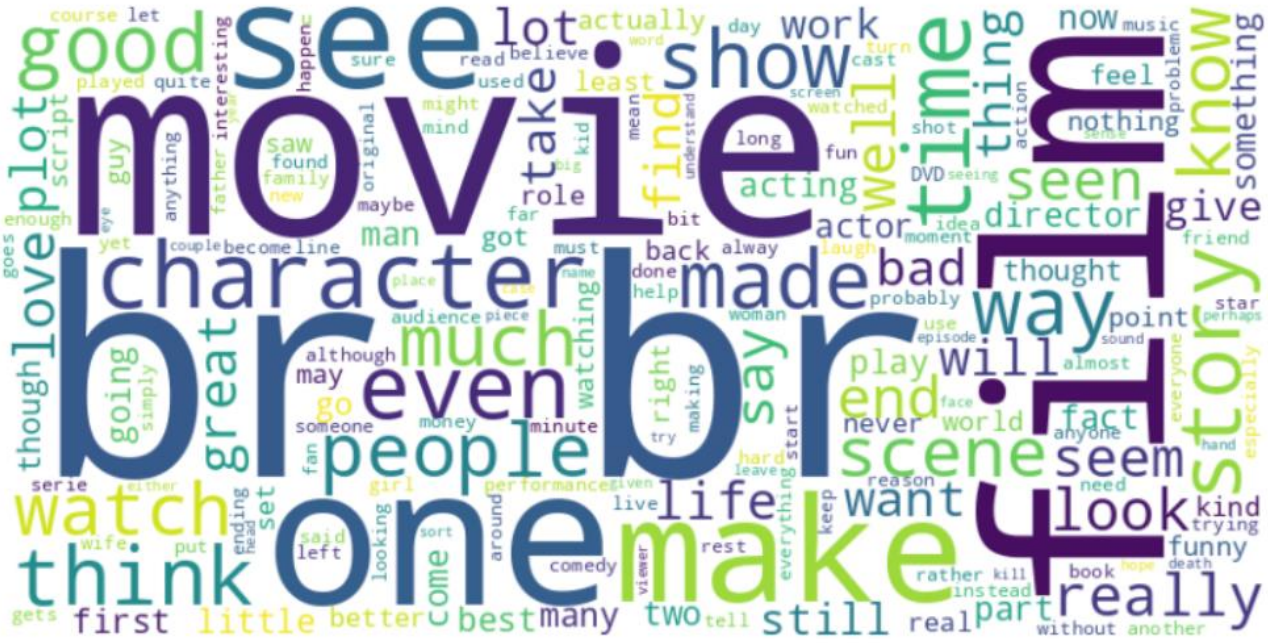
- **Problem Statement**: The task is to analyze sentiment in text data (e.g., movie reviews) and classify them as positive or negative.

- **Objective**: Develop and evaluate a sentiment analysis pipeline, comparing baseline and advanced methods.

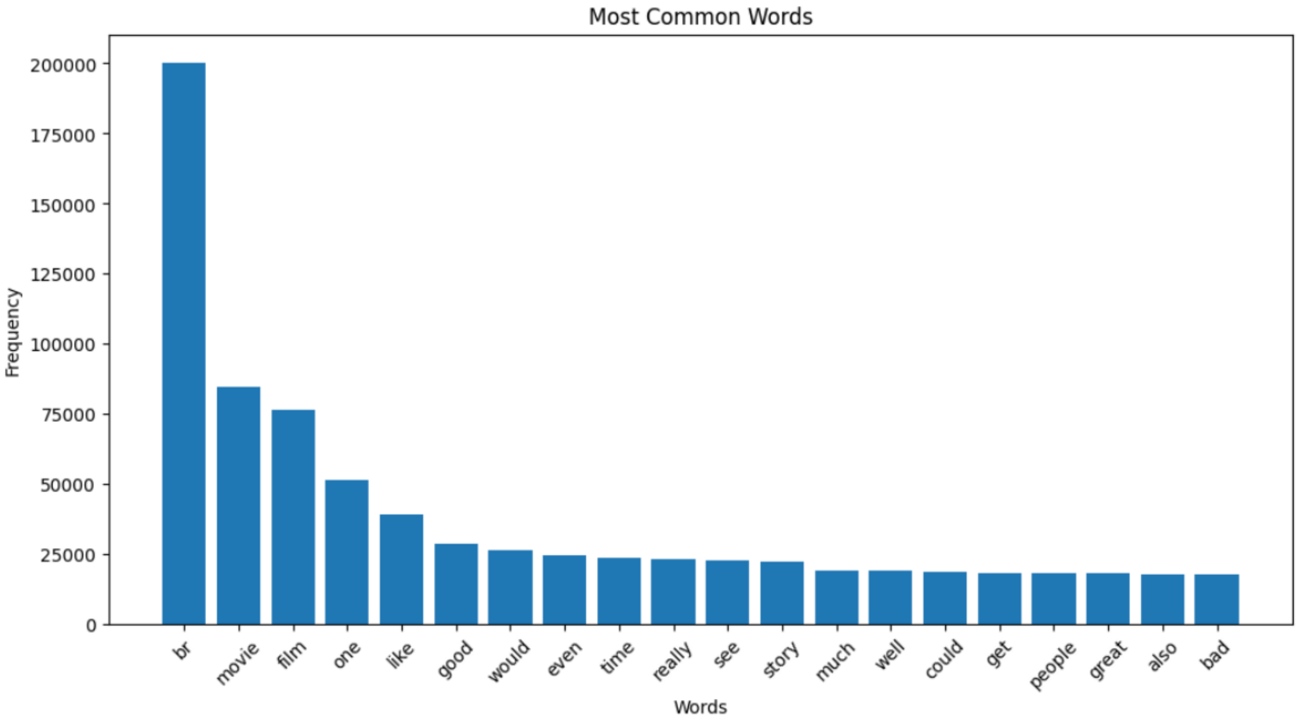
Data Description

- **Source**: IMDB movie reviews dataset.
- **Instances**: 50,000 reviews.
- **Features**: Text of reviews and sentiment labels.
- **Splits**: Data is divided into training (80%) and testing (20%) sets.

Initial exploration of the data showed that the average review word count is 231 words with a minimum of and maximum of 2,470 words. The following visualization shows the most frequent words in the dataset:



The following is a histogram showing the frequency of the most common words in the dataset:



Which shows that the word “br” has to be cleaned as it is an HTML noise in the data

Baseline Experiment

The goal of the baseline experiment is to develop a simple sentiment analysis pipeline using traditional NLP techniques and evaluate its performance using the following steps:

1) **Data Cleaning and Preparation**
   This included removing duplicates, cleaning and normalizing the data by removing HTML tags, and converting to lowercase.

2) **Feature Extraction and Model Training**
   - **TF-IDF Vectorization**: Converting text data into numerical features using TF-IDF vectorization to capture the importance of words in the documents.
   - **Logistic Regression**: Using logistic regression as the classification algorithm to predict sentiment labels.

3) **Evaluation**
   The model achieved an 89.5% accuracy with a 0.9 F1 score which reflects a good score.

Advanced Experiment

The goal of the advanced experiment is to enhance the baseline sentiment analysis pipeline by incorporating additional text preprocessing steps like the following:

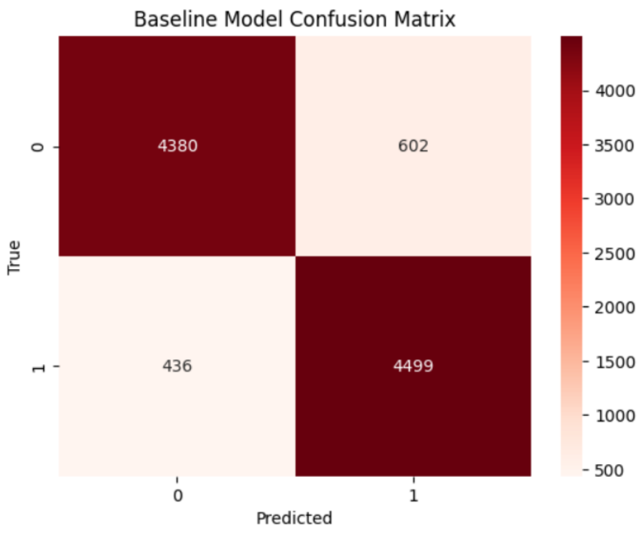
1. **Enhanced Preprocessing:**
   - Negation Handling: Modify the tokenization process to handle negations more effectively.
   - Removing Non-alphabetic characters
   - Removing stop words
   - Stemming: Apply stemming to reduce words to their root forms.

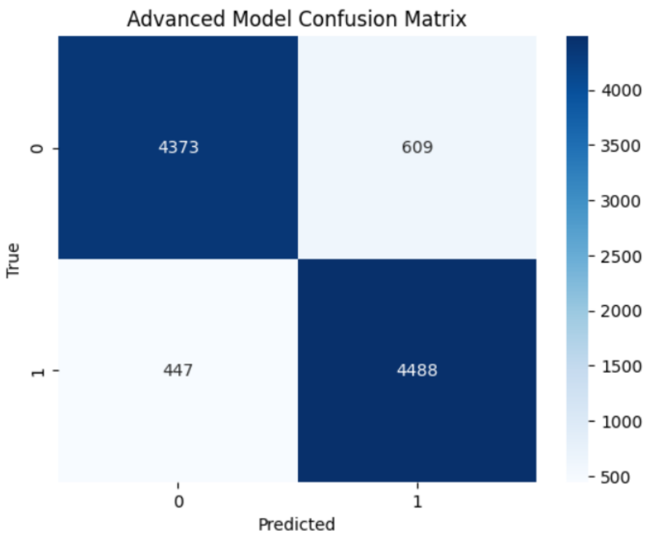
2. **Feature Extraction and Model Training**
   Using the same TF-IDF Vectorization and Logistic regression classifier

3. **Evaluation**
   The model achieved an 89.3 accuracy and 0.89 F1 score which is not very different than from the baseline model

Overall Conclusion

Comparing the 2 models with each other, generally the advanced model will be more helpful as it deals with more logical problems like negation handling better than the baseline model. Following is the confusion matrix of both models which are not very different from each other, but logically the advanced one is better.





Reflection Questions

- **Biggest Challenge**
  Data cleaning was the most challenging task, it included handling extensive preprocessing tasks like removing noise, normalizing text, and dealing with negations which was complex and time-consuming.

- **Valuable Insights**
  I noticed that advanced techniques like word embeddings capture semantic meaning better than traditional methods and effective preprocessing significantly improves model performance.

How to use

A classification pipeline was designed at the end of the notebook to easily classify any new input without running the whole notebook just follow the following steps:

If you are using the Kaggle Notebook:

1- Make sure the path of the uploaded model weights is correct in the first cell of the classification pipeline.

2- Put your text in the *new_input* variable of the last cell.

3- Run all cells starting from the classification pipeline section

If you are using any other environment:

Please make sure to upload the model weights and adjust the model path then continue with the previously described steps