Positive vs. Negative Sentiment Classification: A Baseline and Improved Pipeline

Nur Adila Hanis Binti Wahab (2024148035)

1 Introduction

In this project, I focus on a text classification task that aims to classify IMDb movie reviews as either positive or negative. A naive baseline model is first implemented, followed by an improved pipeline that leverages an existing pre-trained model. Both approaches are evaluated using two metrics which are accuracy and F1 score. In addition, four qualitative examples are provided to further analyze model performance. Finally, reflections on the approaches are discussed, along with their limitations

2 Task Definition

- **Task description:** This project focuses on training a text classification model to accurately identify positive and negative sentiment in movie reviews.
- **Motivation:** This task is particularly interesting because classifying comments is often subjective. Humans and computers may interpret the same text differently, leading to different conclusions about its sentiment. While humans can usually recognize whether a comment is genuinely positive or expressed sarcastically, this distinction can be challenging for AI models. Therefore, this project aims to explore the ability of AI and modern technologies to correctly distinguish between these types of text. This curiosity is the main reason why this topic was choosen.
- **Input / Output:** The model will load some data sets from Internet Movie Database (IMDb) website. Therefore, it will see some of the reviews text and scan them to determine either it is a positive or a negative review. The code will produce 0 for negative and 1 for positive.
- **Success criteria:** For this project, the system is considered successful if it meets the following criteria:
  1. Model performance:
     The system should achieves a strong performance result using standard evaluation metrics such as the accuracy, and F1 Score.
  2. Improvement over baseline:
     The system needs to show improvement compared to the baseline model

3 Methods

For this project, I used both naive baseline and improved AI pipeline methods

3.1 Naïve Baseline

Naíve Baseline Method: Bag-of-Words + Logistic Regression

Method Description

The baseline model uses the Bag-of-Words (BoW) approach with Logistic Regression. First, the input text will be turn into a vector of word counts through the CountVectorizer(). BoW has no understanding of the sentence meanig. It only counts word frequencies and ignores grammar, context and the word order. Next, the logistic regression classifier will learns which words indicate positive or negative sentiment. For example, the word 'amazing' has a positive weight while 'boring' has a negative weight. Finally, when given a new review, it will sums up the weighed word counts and outputs 1 for positive and 0 for negative sentiment.

Why naïve

This baseline is naive because it ignores text context as it sees each word independently. Meaning that some expressions such as "pretty bad" or "not bad" cannot be understand as how humans perceive them. In additon, this baseline also lacks critical analysis as it does not adapt to nuanced expressions or the tone. Since the vocabulary is fixed, new words during testing will become unknown and the model could not correctly evaluate them. This model is also sensitive to spelling. Therefore, it cannot evaluate them.

Likely Failure Modes

This baseline is expected to fail or perform poorly in these situations:

1. Negation
   Sentences using negation such as "not like" or "not good" may be classified as positive because of the word "like" and "good"

2. Sarcasm
   Since the model ignores the context and mainly depends on each word independently, it may think it is positive.

3.2 AI Pipeline

AI Baseline Method: TF-IDF vectorizer + Linear Support Vector Machine(LinearSVC) classifier

Models Used

I used a NLP pipeline consisting of a TF-IDF vectorizer and a Linear Support Vector Machine (LinearSVC) classifier. The TF-IDF component converts text into weighted numerical features, while the LinearSVC model serves as the decision-making classifier. This combination is widely used for sentiment analysis tasks because it is efficient and performs well on various text data.

Pipeline stage

1. Preprocessing
   The input text is cleaned and tokenized automatically by the TF-IDF vectorizer. Common Englishbstopwords are removed, and the text is converted into lowercase. Unigram and bigram feature is also enabled to allow the model to capture both single words and short phrase such as "not good" or "very bad", which are important for sentiment detection.

2. Embedding or Representation
   TF-IDF converts each document into a numerical vector where terms are weighted by their importance in the review and across the dataset. The feature size is limited to 5000 to keep the representation efficient.

3. Decision/ Ranking component
   A linearSVC classifier uses the TF-IDF vectors to leanr a boundary between positive and negative reviews. It assigns positive weights to terms associated with positive sentiment and negative weights to terms linked to negative sentiment.
4. Post-Processing
   No post-processing is needed because the SVM directly outputs the predicted class label.

Design Choices and Justification
I selected the TF-IDF + LinearSVC pipeline because it is simple and offers a good performance. Unlike the bag-of-words counts, TF-IDF captures the importance of term more effectively. In additon, using the bigrams enables the model to detect short sentiment phrases. Linear SVC is a robust and widely used classifier for text data which perform better than logistic regression overall. This pipeline is computationally lightweight, trains quickly on small datasets, and provides a strong baseline before moving to larger transformer-based models.

4 Experiments

4.1 Datasets

Dataset Description

- **Source:** The dataset is derived from the IMDb movie review dataset, a widely used benchmark for sentiment analysis. It contains human-written reviews labeled as either positive or negative.
- **Total examples:** For this project, I used 200 samples. 100 for positive and 100 for negative.
- **Train/Test split:** I divided the data into 150 for training (75 positive and 75 negative) and 50 for testing (25 positive and 25 negative)
- **Preprocessing steps:** Text is lowercased, tokenized, and converted intp TF-IDF features. English stopwords are removed, and both unigrams and bigrams are included.

4.2 Metrics

For this project, I used two metrics which are accuracy and F1 score to evaluate both baseline and AI pipeline. Accuracy and F1 score are used because they provide a complementary perspectives on model performance. Accuracy measures overall correctness and offers a general overview. Meanwhile, F1 Score highlights the model's robustness in handling classification errors. Together, these metrics give a reliable and balanced evaluation of the text classification models.

4.3 Results

Result:

Method	**Accuracy**	**F1 Score**
Baseline	0.6800	0.6667
AI Pipeline	0.7000	0.7368

Qualitative Examples:

Text	**True Label**	**Baseline**	**AI Pipeline**	**Comment**
”This movie was extremely boring”	Negative (0)	0	0	Both model correct
”I really love this movie”	Positive (1)	0	1	AI Pipeline corrected the baseline mistake
”Great! another sequel no one asked for”	Negative (0)	0	1	AI pipeline made a mistake while baseline was correct
”Confusing at first, but very enjoyable!”	Positive (1)	0	1	AI pipeline corrected the baseline mistake

5 Reflection and Limitations

Overall, the AI pipeline model performed better than expected, particularly in capturing clear positive and negative sentiment compared to the naive baseline. The use of a pre-trained model significantly improved both accuracy and F1 score. However, the model still struggled with ambiguous reviews, especially those containing sarcasm or mixed sentiment, which remained difficult to classify correctly. Accuracy and F1 score captured overall performance reasonably well, but they did not fully reflect qualitative issues such as misinterpreting sarcasm. As a result, some predictions that were scored as correct by the metrics still felt incorrect when reviewed qualitatively. With more time, I would like to experiment with larger models, additional fine-tuning epochs, and a more advanced techniques. I would also like to try a different type of task and explore more about this fine-tuning.

References

1. IMDb Large Movie Review Dataset (Maas et al., 2011)

2. TfidfVectorizer — scikit-learn documentation https://scikit-learn.org/stable/modules/generated/sklearn.feature_extraction.text.TfidfVectorizer.html

3. LinearSVC — scikit-learn documentation https://scikit-learn.org/stable/modules/generated/sklearn.svm.LinearSVC.html

4. Bag-of-Words scikit-learn developers. (n.d.). Text feature extraction. scikit-learn documentation. https://scikit-learn.org/stable/modules/feature_extraction.html

5. Metrics, https://scikit-learn.org/stable/modules/model_evaluation.html