COM3110: Sentiment Analysis of Movie Reviews

Samiha Fansur

1 Implementation details

1.1 Prior Calculation

The prior probabilities for each sentiment class are computed in the `MultinomialNaiveBayes` class within the `calc_all_prior_probabilities()` method, as shown in the file `multinomial_naive_bayes.py` (lines 23-43). This method calculates the probability of each class based on the distribution of sentiments in the training dataset. The `calc_prior_probability()` function determines the proportion of words for a given sentiment relative to the total number of words.

1.2 Likelihoods Calculation

Likelihoods are calculated in the same `MultinomialNaiveBayes` class using the `calc_all_likelihood()` method (lines 45-122). The method employs the `calc_all_relative_likelihoods()` function to determine the probability of each token within a class, adjusting for Laplace Smoothing. This ensures that new features encountered in the test dataset do not result in probability issues.

1.3 Feature Extraction

Feature extraction is handled by several classes, each tailored to a specific method. The `FeatureExtraction` class, as defined in `feature_extraction.py`, encompasses different techniques for identifying and selecting the most informative features from the dataset. This class includes the `featureExtraction()` method (lines 86-130), which selects informative features based on the sentiment distribution in the dataset using the Sentiment bias extraction method.

Additionally, the `SentimentBias` class, located in `sentiment_bias.py` (lines 25-118), encapsulates the sentiment bias calculation which weighs the sentiment distribution across the dataset. By analysing the frequency and distribution of tokens in different sentiment classes, this class helps in selection features that are most likely to contribute to the accurate classification of sentiments.

1.4 Macro-F1 Score

Model's performance is based on the macro-F1 score, calculated by the `Evaluation` class's `macro_f1_score()` method in `evaluation.py` (lines 64-100). This method aggregates individual F1 scores computed for each class label, taking into account true positives, false positives, and false negatives.

1.5 Pre-processing Steps

The model employs a series of pre-processing steps implemented in the `PreprocessingData` class within `preprocessing_data.py` (lines 29-91). These steps include lowercasing (`tokens_to_lowercase()`) and stopwords removal (`filter_stopwords()`) implemented to reduce noise and improve the interpretive accuracy of the model.

1.6 Code Performance

Using Python's `defaultdict` for counting, `numpy` for vectorized operations, and `pandas` for data frame manipulation, ensures fast and memory-efficient computations. Object-oriented design principles and modular design has also been followed to enhance maintainability and scalability of the code. Efficient data structures, numpy for array operations, pandas for data handling, and efficient looping constructs have also been implemented.

2 Feature Selection

2.1 TF-IDF

TF-IDF stands as an advanced alternative to count vectorization by mitigating the influence of commonly occurring words across documents. TF-IDF weighs terms based on their importance, which is cruical for distinguishing nuanced sentiment expressions in the 5-value classification [1]. Nevertheless, TF-IDF may not capture the semantic orientation of words, a limitation in the context of sentiment analysis.

2.2 Sentiment Bias

Sentiment bias quantifies the tendency of words to associate with particular sentiment classes. It is a sophisticated method that accounts for the distributional skewness of words in sentiment-laden corpora [2]. Sentiment bias can capture subtle sentiment nuances but may be less effective for words with neutral or ambiguous sentiment associations. I calculated and ordered the sentiment bias for each word, then selectively used a percentage of the top features. For the three-class model, using 97% of feature tokens was optimal, whereas, for the five-class model, the best results were obtained with 38% of the feature tokens. This approach highlights the method's flexibility and precision in adapting to the specific needs of sentiment analysis tasks.

2.3 Sentiment Lexicon

The sentiment lexicon approach employs predefined lists of positive and negative words to ascertain sentiment. It is predicated on the hypothesis that certain words have inherent sentiment values, making it a robust method for capturing explicit sentiment expressions [3]. Its limitation lies in its binary classification of words, potentially overlooking the spectrum of sentiment intensities.

2.4 Method Combination

Combination of methods, specifically TF-IDF and sentiment lexicon, aimed to enhance the classifier's discernment between closely related sentiments, as evidenced by empirical results [4]. However, this combination increases computational complexity and may introduce noise through redundant features.

In conclusion, the feature selection aims to balance the trade-off between feature expressiveness and classifier efficiency. TF-IDF adds term significance, sentiment lexicon captures explicit polarities, and sentiment bias reflects distributional preferences.

3 Results and Discussion

Preprocessing	Method	3-class all_words Score	3-class feature Score	3-class Runtime(s)
p1+p2	m1	0.504596	0.472100	5.45
	m2	0.504596	0.490034	3.98
	m3	**0.504596**	**0.506109**	**3.60**
	m1 + m2	0.504596	0.500235	6.65
p1+p2+p3+p4	m1	0.506617	0.470000	8.00
	m2	0.506617	0.480000	7.25
	m3	0.506617	0.494621	7.00
	m1 + m2	0.506617	0.482000	8.50

Table 1: 3-class Macro-f1 scores under various preprocessing and feature selection methods)

Preprocessing	Method	5-class all_words Score	5-class feature Score	5-class Runtime(s)
p1+p2	m1	0.300958	0.370856	6.03
	m2	0.231221	0.401556	3.43
	m3	0.231221	0.411403	3.12
	m1 + m2	0.231221	0.410089	7.44
None	m1	0.231221	0.370856	7.23
	m2	0.300958	0.381155	5.59
	m3	**0.300958**	**0.412846**	**3.52**
	m1 + m2	0.300958	0.390143	8.21

Table 2: 5-class Macro-f1 scores under various preprocessing and feature selection methods)

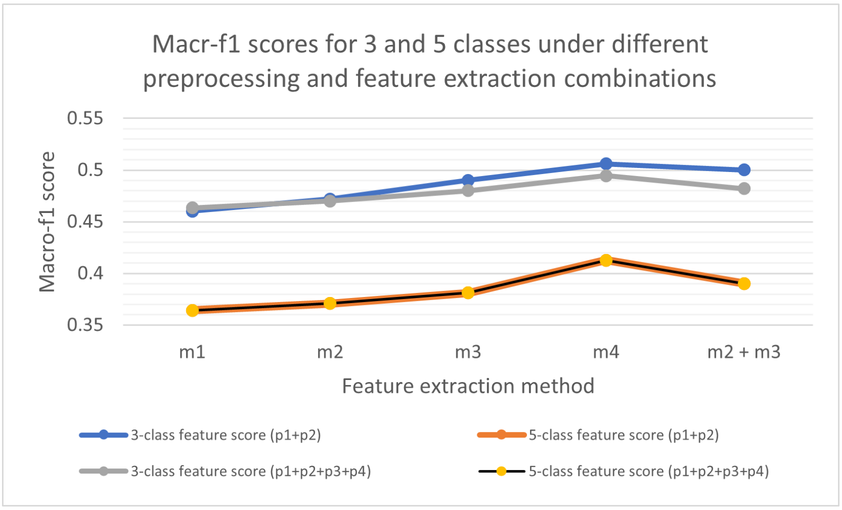

Figure 1: Macro-F1 scores with varied preprocessing and feature selection

Label	Detail
p1	Lowercasing
p2	Stopword Removal
p3	Lemmatization
p4	Stemming
m1	TF-IDF
m2	Sentiment Lexicon
m3	Sentiment Bias

Table 3: Preprocessing and feature extraction methods

The p1 + p2 m3 model, uses sentiment bias, consistently outperforming other feature extraction methods, marking it as the best model. It achieves macro-F1 scores of 0.506109 and 0.412846, beating the 'all words' scores of 0.504596 and 0.300958 for the 3-class and for the 5-class sentiment analysis respectively. No preprocessing is applied to 5-class as it consistently decreases its performance but enhances the 3-class models performance.

TF-IDF, while effective for highlighting unique terms in documents for search algorithms, encounters limitations in sentiment analysis. Its emphasis on term rarity and uniqueness can misrepresent the significance of words with respect to sentiment. For instance, a neutral but rare word like 'giraffe' might receive a high TF-IDF score, contributing little to sentiment analysis. Conversely, common neutral words like 'movie' are scaled down in TF-IDF, aligning with their sentiment neutrality. However, the key challenge with TF-IDF in sentiment analysis lies in its focus on term uniqueness rather than emotional or opinionated weight, which is central to sentiment analysis. This can lead to misrepresentations where the importance of words in sentiment terms is not accurately captured by their TF-IDF weights.

Stemming and lemmatization, while beneficial for normalising lexical variations, prove less advantageous for the 3-class model. They potentially obscure sentiment distinctions by reducing words to their base forms. However, their impact is neutralised within the 5-class model, which can still differentiate sentiment despite the words' reduced forms.

The confusion matrices below offer a clear visualisation of the sentiment analysis model's performance for both 5-class 3 and 3-class 2 models. They effectively highlight the true and false predictions, guiding improvements in the model. For instance, upon observing a high number of false negatives in the confusion matrix, I adjusted the classification thresholds, which improved the results.

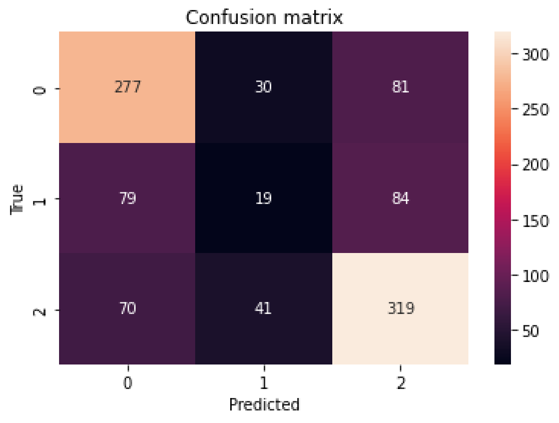

Figure 2: Confusion Matrix for the Best Model (p1 + p2 m3) in 3-Class Sentiment Analysis

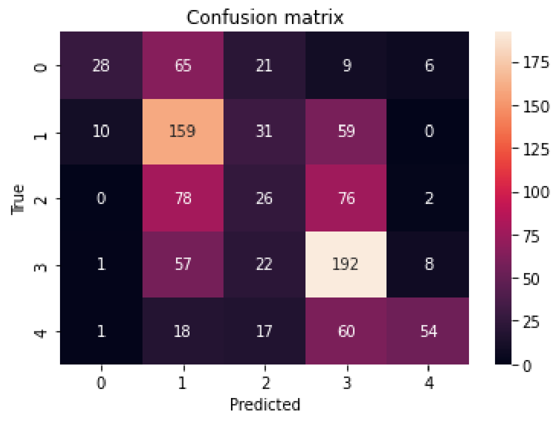

Figure 3: Confusion Matrix for the Best Model (p1 + p2 m3) in 5-Class Sentiment Analysis

In conclusion, the sentiment bias model with lowercasing and stopwords removal (p1+p2 m3) is identified as the most effective for this sentiment analysis. It adeptly captures the complexities of sentiment expression and avoids the pitfalls of other methods that either overemphasise neutral terms or oversimplify the sentiment spectrum.

3.1 Model Performance and Limitations

Optimal performance for the 3-class model was achieved with 97% of the feature tokens, whereas for the 5-class model it was with 38%. The highest Macro-F1 scores produced were 0.506109 for the 3-class model and 0.412846 for the 5-class model. Moreover, the model's key limitation is due to treating features as independant features, thus reducing performance in capturing the relational dynamics of words in movie reviews.

4 Error Analysis

ID	**Actual**	**Predicted**	**Reason for Failure**	**Model Improvements**
2083	2	0	Dual-meaning phrase misinterpreted due to lack of contextual analysis.	Incorporate bi-directional context parsing to understand phrases with multiple meanings.
4241	0	2	Misled by isolated positive words without considering overall negative sentiment.	Utilise n-grams to maintain the integrity of sentiment expressions and contextual embeddings.
1757	2	0	Irony in proper nouns and implied sentiment not captured.	Enhance the model with named entity recognition and sentiment-specific word embeddings.
7231	0	2	Failure to reconcile contrasting sentiments within the sentence.	Implement sentiment compositionality analysis to evaluate how different sentence parts contribute to overall sentiment.

Table 4: Analysis of Errors and Proposed Improvements for a 3-class Model

ID	**Actual**	**Predicted**	**Reason for Failure**	**Model Improvements**
479	4	0	The model fails to capture the positive sentiment due to the absence of sentiment-laden adjectives after preprocessing.	Adapt preprocessing to preserve adjectives and adverbs which often carry significant sentiment value.
7335	3	0	This sentence exemplifies the model's struggle with implied sentiments, where the sentiment is not directly stated but suggested.	Integrate a context-aware feature that can detect and interpret implied sentiments.
4164	4	1	The model undervalues the sentence sentiment, overlooking the intensity conveyed by 'passionately.'	Implement a feature to identify and score sentiment-modifying adverbs appropriately.
3201	0	4	Here, a neutral term ('movie') is overvalued, while a negative phrase ('paint-by-numbers') is not given due weight.	Refine the model's feature weighting mechanism to better handle idiomatic negations and neutral terms within a sentiment context.

Table 5: Analysis of Errors and Proposed Improvements for a 5-class Model

References

[1] J. Ramos, “Using tf-idf to determine word relevance in document queries,” in *Proceedings of the First Instructional Conference on Machine Learning*, 2003.

[2] A. Severyn and A. Moschitti, “Unitn: Training deep convolutional neural network for twitter sentiment classification,” in *Proceedings of the 9th International Workshop on Semantic Evaluation (SemEval 2015)*, 2015.

[3] M. Hu and B. Liu, “Mining and summarizing customer reviews,” in *Proceedings of the Tenth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, 2004.

[4] S. Wang and C. D. Manning, “Baselines and bigrams: Simple, good sentiment and topic classification,” in *Proceedings of the 50th Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2012.