Artificial Course Assignment

Author Wang Xi

Student Number 6243112053

Abstract

This project leverages the AI techniques studied in this semester's "Advances in Artificial Intelligence" course, focusing on applying Naive Bayes algorithms to sentiment analysis using the IMDB movie review dataset. The course provided foundational insights into search algorithms, probabilistic models, and machine learning methods, which shaped our approach to this project.

For the task, I built individual classifiers—Multinomial, Bernoulli, and Complement Naive Bayes—along with Logistic Regression, combining them in a soft-voting ensemble model to enhance classification accuracy. Key hyperparameters, including the alpha for Naive Bayes, were optimized, and the final ensemble achieved an accuracy of 89% with a strong balance of precision and recall across sentiment classes.

Our results, visualized through confusion matrices and F1 scores, confirm that the ensemble approach is robust in identifying positive and negative sentiments, making it well-suited for real-world applications in sentiment analysis and text classification.

1. Introduction

1.1. A summary of course

Artificial Intelligence (AI) is a transformative field of study focused on developing machines capable of exhibiting intelligence similar to that of humans. Through computer programs and algorithms, AI aims to simulate and extend various aspects of human cognition, including understanding, learning, reasoning, problem-solving, perception, and language comprehension. This technology is revolutionizing industries such as healthcare, finance, transportation, and manufacturing by driving automation, improving efficiency, and enabling new capabilities that were previously impossible. In this semester's "Advances in Artificial Intelligence" course, I had the opportunity to learn a variety of AI techniques that are both practical and innovative.

In the chapter on search algorithms, I explored fundamental search techniques like Depth-First Search (DFS) and Breadth-First Search (BFS), which provide foundational knowledge for navigating problem spaces. Building on this, we also studied advanced methods like A* search, Greedy best-first search, and heuristic-driven approaches that enhance efficiency by focusing the search process toward optimal solutions. This chapter not only covered theoretical aspects but also offered practical applications, such as solving classic problems like the 8-puzzle and the 8-queen problem, both of which demonstrate the effectiveness of search algorithms in structured problem-solving environments.

In later chapters, I learned about Alpha-Beta pruning, which optimizes decision-making algorithms in games like tic-tac-toe by reducing the number of nodes evaluated in a search tree. I also implemented the Minimax algorithm, which is essential for AI-based game playing. Additionally, we covered machine learning fundamentals, including linear regression—a classic model that provided insights into supervised learning. I learned about its cost function, optimization techniques, and how linear regression can be applied to predict continuous outcomes, laying the groundwork for more complex predictive models. Of all the topics, Bayesian networks intrigued me the most. Bayesian networks are a type of probabilistic model that represents dependencies among variables and offers a framework for reasoning under uncertainty. With intuitive visual representations and solid performance in handling uncertainty, they are particularly useful in real-world scenarios where data may be incomplete or noisy. By studying and implementing Bayesian networks, I enhanced both my understanding of probabilistic models and my programming skills.

1.2. My project introduction

For this project, I chose to work on sentiment analysis of IMDB movie reviews, a classic yet highly relevant problem in the field of natural language processing (NLP). Sentiment analysis is the process of classifying text data into categories like positive, negative, or neutral, based on the sentiment expressed. Movie reviews are particularly useful for this analysis because they tend to be opinion-rich and come with a natural label, making them ideal for supervised learning. In today's digital world, people express opinions about everything from products and services to movies and political issues through online platforms. Analyzing and summarizing public opinion automatically can provide valuable insights to businesses, researchers, and policymakers. In the movie industry specifically, sentiment analysis of reviews can reveal trends, audience preferences, and the general reception of films. This is crucial for marketing strategies, targeted content recommendations, and decision-making processes in film production and distribution.

1.3. Why choose this project

Sentiment analysis of large datasets like IMDB reviews enables us to explore machine learning and NLP techniques on real-world, noisy data, enhancing the robustness and accuracy of our models. Given that the field of NLP is rapidly advancing, sentiment analysis also offers a chance to test new algorithms and ensemble methods, assessing their effectiveness in a practical scenario.

I chose this problem because it aligns closely with my interest in language processing and its applications in understanding human emotion and opinions through technology. Sentiment analysis also provides a foundational task that allows for experimenting with various machine learning techniques, including ensemble learning, which was covered in our AI course. Ensemble learning, which combines the strengths of different models, is particularly exciting for text classification tasks, where a single algorithm may not capture the complexity of language as effectively as multiple models.

The real application of this problem extends beyond movie reviews. Businesses can use similar sentiment analysis models to evaluate customer feedback, identify pain points in services, or understand public sentiment towards a brand. Additionally, sentiment analysis tools can be applied in fields like finance for stock market predictions based on news sentiment, in politics for gauging public opinion, or even in healthcare to assess patient feedback on medical care.

2. Relate Work

For the problem tackled in this project, sentiment analysis of text data, several existing approaches are commonly used to classify sentiment with varying levels of effectiveness, complexity, and application scenarios. Below are descriptions of the primary existing methods, their strengths, and their limitations.

2.1. Traditional Machine Learning Methods

Traditional machine learning algorithms like Naive Bayes, Support Vector Machines (SVM), and Logistic Regression are widely used for text classification, including sentiment analysis. These models are generally paired with feature extraction techniques such as Bag of Words (BoW) or Term Frequency-Inverse Document Frequency (TF-IDF) to represent text in a numerical format suitable for machine learning algorithms.

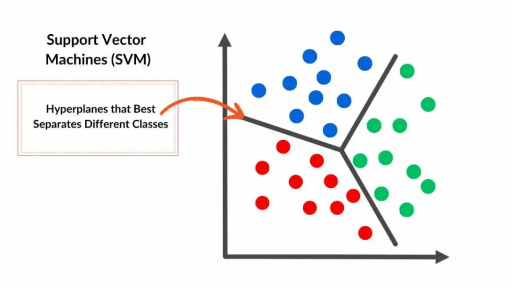
Support Vector Machines (SVM)
Hyperplanes that Best Separates Different Classes

Figure 1. SVM Model

- Advantages: These models are relatively easy to understand and implement. They work well for smaller datasets and have lower computational requirements compared to deep learning approaches. They are also effective when paired with carefully engineered features, as they can be fine-tuned to classify sentiments accurately within specific domains.
- Disadvantages: They rely heavily on feature engineering, which can be labor-intensive and requires domain expertise. These models typically struggle with capturing the contextual and sequential relationships in text, as they treat words independently, leading to suboptimal performance in understanding complex linguistic structures.

2.2. Deep Learning Models

With the advent of deep learning, Recurrent Neural Networks (RNNs) and Long Short-Term Memory (LSTM) networks have been widely adopted for text-based tasks.[1]These models can capture sequential dependencies within text, allowing them to perform better than traditional machine learning methods in sentiment analysis. Convolutional Neural Networks (CNNs) have also been adapted for text classification tasks, as they can detect specific patterns within word sequences.[2]

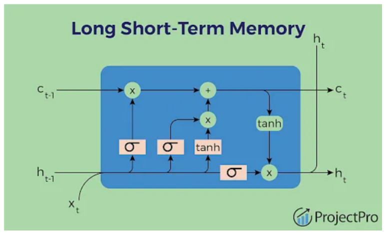

Figure 2. LSTM Model

- Advantages: Deep learning models are capable of capturing context and word order information due to their inherent sequential processing capabilities. They typically require less manual feature engineering because they can learn hierarchical features automatically.
- Disadvantages: Deep learning models are computationally expensive and often require large datasets to achieve optimal performance. Training deep networks can be time-consuming, and they tend to overfit if not carefully regularized or trained with sufficient data.

2.3. Transformers and Pretrained Language Models

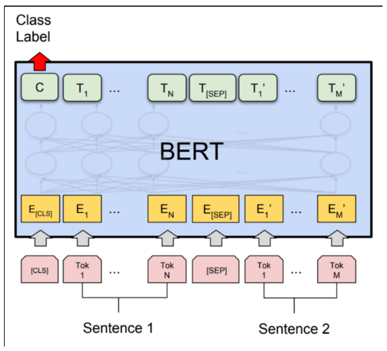

Figure 3. BERT Model

The use of transformer-based models, such as BERT (Bidirectional Encoder Representations from Transformers), has recently advanced the state-of-the-art in natural language processing.[3]These models leverage bidirectional attention mechanisms, allowing them to understand the context of a word based on both its preceding and following words.[4]

- Advantages: Transformers capture context more effectively than traditional RNNs and LSTMs, as they consider bidirectional information across the text sequence.[5] Pretrained models like BERT come with prior knowledge from training on massive datasets, which improves their performance on tasks like sentiment analysis even with smaller domain-specific data.
- Disadvantages: They are highly resource-intensive, requiring GPUs or TPUs for efficient training and inference. Fine-tuning transformers on domain-specific tasks can be complex and computationally demanding, especially for practitioners with limited hardware resources.

2.4. Conclusion

Overall, each method presents unique benefits and limitations depending on the problem's scale, data availability, and computational resources. The choice of model often reflects a balance between performance requirements and practical considerations, such as computational efficiency and ease of implementation.

2.5. Illustrations, graphs, and photographs

All graphics should be centered. Please ensure that any point you wish to make is resolvable in a printed copy of the paper. Resize fonts in figures to match the font in the body text, and choose line widths which render effectively in print. Many readers (and reviewers), even of an electronic copy, will choose to print your paper in order to read it. You cannot insist that they do otherwise, and therefore must not assume that they can zoom in to see tiny details on a graphic.

3. My Approach

In this project, I chose to implement a Bayesian network for sentiment analysis. A Bayesian network is a probabilistic model used to represent conditional dependencies among variables. It captures uncertainty and dependency structures within data by decomposing a complex problem into multiple conditionally independent subproblems. This makes Bayesian networks well-suited for sentiment analysis, as they can capture underlying sentiments in the text via probabilistic relationships.

3.1. Algorithm and Model Details

The Bayesian network used in this project is built on Bayes' theorem, which provides a foundation for probabilistic inference by calculating the likelihood of an event based on prior knowledge. Bayes' theorem is expressed as:

$$P(C|X) = \frac{P(X|C) \cdot P(C)}{P(X)}$$

where $P(C|X)$ represents the posterior probability of class C given the features X, $P(X|C)$ is the likelihood of observing features X given class C, $P(C)$ is the prior probability of class C, and $P(X)$ is the probability of the features X. This formula enables the computation of the joint probability of different features in text data, allowing us to predict the likelihood of the input belonging to a specific class.

For this project, I implemented several classical Bayesian classifiers: Multinomial Naive Bayes (MultinomialNB), Bernoulli Naive Bayes (BernoulliNB), and Complement Naive Bayes (ComplementNB). These classifiers are simplified Bayesian networks that assume independence between features, a core assumption in Naive Bayes models. This assumption allows the computation to be efficient, making Naive Bayes well-suited for high-dimensional data like text.

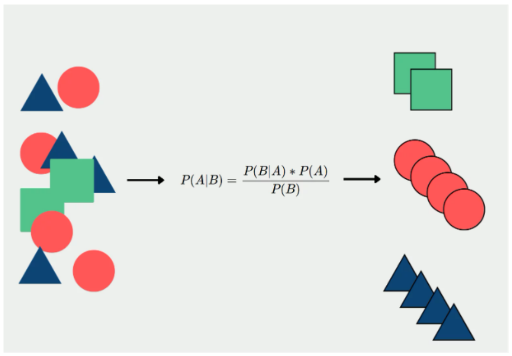

Figure 4. Bayes Formula

The TF-IDF (Term Frequency-Inverse Document Frequency) method was employed to extract text features. TF-IDF transforms text data into numerical representations by calculating the frequency of terms within each document (Term Frequency) and adjusting for how commonly the terms appear across all documents (Inverse Document Frequency). The formula for TF-IDF is:

$$\text{TF-IDF}(t, d) = \text{TF}(t, d) \times \log\left(\frac{N}{\text{DF}(t)}\right)$$

where $\text{TF}(t, d)$ is the frequency of term t in document d, N is the total number of documents, and $\text{DF}(t)$ is the number of documents containing term t.

The process begins with data preprocessing, which includes removing stop words and converting all text to lowercase for normalization. The processed text data is then transformed into a bag-of-words model using TF-IDF to emphasize the importance of specific terms. After feature extraction, I trained the Bayesian models on the preprocessed data and used them to make predictions on the test set. This approach leverages the simplicity and interpretability of Bayesian classifiers, combined with the effectiveness of TF-IDF, to deliver efficient and accurate text classification results.[6]

3.2. Why I Chose This Algorithm

Bayesian classifiers are widely used in text classification because they are fast, stable, and perform well even on smaller datasets. Compared to complex deep learning models, Bayesian classifiers require less data and computational resources while delivering strong performance. Furthermore, Bayesian network models offer a high level of interpretability, making prediction results easier to understand, which is critical in sentiment analysis. Since Bayesian networks can infer sentiment tendencies based on probability, they excel in handling issues such as imbalanced samples.

3.3. My improvements

In this project, I optimized the traditional Bayesian model in several ways to improve performance:

- Parameter Tuning: I adjusted the smoothing parameter, alpha, to optimize model precision and recall. A well-tuned smoothing value reduces overfitting and improves classification performance.
- Feature Extraction Optimization: During the TF-IDF feature extraction phase, I optimized the n-gram range. Using an n-gram range of (1,3) allows the model to capture combinations of words and their neighboring terms, enabling better contextual understanding.
- Ensemble Approach: By integrating multiple Bayesian models (MultinomialNB, BernoulliNB, and ComplementNB) through hard voting, I enhanced the stability and accuracy of the classifier.

4. Experiments

4.1. Description of the Data and Linkage to the Dataset

For this project, I used the IMDB Movie Reviews Dataset, which consists of 50,000 movie reviews labeled as either positive or negative, with 25,000 reviews for training and 25,000 for testing. The dataset contains movie reviews written in natural language, with each review labeled as either ”positive” or ”negative.” The goal of the project is to apply Bayesian network-based classifiers to predict the sentiment of these reviews.[4]

- Dataset Size: 50,000 reviews (25,000 for training, 25,000 for testing)
- Features: The raw text of movie reviews
- Labels: Sentiment labels (Positive/Negative)

Preprocessing: I used text preprocessing techniques such as removing stop words and applying TF-IDF vectorization to transform the raw text into numerical feature vectors that can be used by machine learning models. The dataset was obtained from a publicly available collection on Kaggle (https://www.kaggle.com/), and I loaded it using pandas for data manipulation and splitting the dataset into training and testing subsets. I then applied TF-IDF vectorization on the text data to convert it into numerical form. The labels were binary (0 for negative, 1 for positive sentiment).

4.2. TF-IDF Vectorization

The Term Frequency-Inverse Document Frequency (TF-IDF) vectorization method is a crucial step in transforming raw text into numerical representations, allowing machine learning algorithms to work effectively with textual data. TF-IDF quantifies the importance of words in a document relative to their occurrence across the entire dataset, making it particularly valuable for tasks like sentiment analysis and text classification.

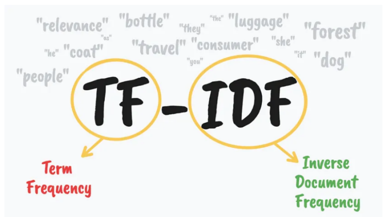
"relevance" "bottle" "they" "luggage" "forest"
"coat" "travel" "consumer" "she" "dog"
"people"
TF-IDF
Term Frequency
Inverse Document Frequency

Figure 5. TF-IDF

The TF-IDF score for a term t in document d is calculated using the formula:

$$\text{TF-IDF}(t,d) = \text{TF}(t,d) \times \log\left(\frac{N}{\text{DF}(t)}\right)$$

where:

- $\text{TF}(t,d)$ is the term frequency, or the count of how often term t appears in document d.
- N is the total number of documents in the dataset.
- $\text{DF}(t)$ is the document frequency, which is the number of documents containing the term t.

By assigning higher weights to terms that are more informative within each document and reducing the impact of terms that are common across many documents, TF-IDF enhances the representation of distinguishing features, helping models focus on words that contribute meaningfully to classifying or identifying the sentiment in text.

During TF-IDF vectorization, English stop words (e.g., "the," "is," "and") were removed. Stop words are common words that appear frequently but often carry little meaning for text classification. By excluding them, the model can concentrate on more informative and content-heavy words that better represent the sentiment or topic of each document. Removing stop words thus reduces noise in the data and improves the clarity of the text features.

I experimented with using n-grams to capture additional context in the text. Specifically, I set the n-gram range to $(1,3)$, which includes unigrams (individual words), bigrams (pairs of consecutive words), and trigrams (triples of consecutive words). This choice allows the vectorization process to consider multi-word expressions, which often convey more specific meanings than single words alone. For instance, in a sentiment analysis task, phrases like "not good" or "very happy" provide crucial context that can influence classification, while individual words like "not" or "good" may be ambiguous on their own.

The inclusion of n-grams was beneficial because:

- **Unigrams** allow the model to learn from individual words and their general sentiment association.
- **Bigrams and Trigrams** capture common expressions, idioms, and contextual phrases that improve the model's ability to discern sentiment based on word combinations.

Through stop word removal and n-gram feature inclusion, the TF-IDF vectorizer produced a feature set that is both refined and contextually rich, enabling the model to better interpret the nuances of each review and make more accurate predictions.

4.3. Model Choices and Hyperparameters

In this project, I employed several Naive Bayes classifiers and an ensemble method to enhance the accuracy and robustness of sentiment classification. Below are the models selected, the rationale behind each choice, and the hyperparameters used.

4.3.1 Multinomial Naive Bayes (MultinomialNB)

The Multinomial Naive Bayes (MultinomialNB) classifier was chosen due to its suitability for discrete features, particularly term frequency data. In this implementation, I set the smoothing parameter $\alpha = 0.1$, which helps to prevent overfitting by smoothing the probability estimates for each feature. The probability $P(w|c)$ of a word w given a class c is calculated as:

$$P(w|c) = \frac{\text{Count}(w,c) + \alpha}{\sum_{w'} \text{Count}(w',c) + \alpha \cdot V}$$

where:

- $\text{Count}(w,c)$ is the number of times word w appears in documents of class c.
- V is the total vocabulary size.
- α is the smoothing parameter, set here to 0.1.

This smoothing helps to handle words that may not appear in every class, thus making the model more generalizable.

4.3.2 Bernoulli Naive Bayes (BernoulliNB)

I also experimented with the Bernoulli Naive Bayes classifier (BernoulliNB), which assumes binary feature values (presence or absence of words) rather than frequency counts. This is particularly useful in text classification where binary indicators of word presence can provide meaningful insights. Similar to MultinomialNB, I set $\alpha = 0.1$ to smooth the probability estimates:

$$P(w|c) = \frac{\text{Count}(w,c) + \alpha}{N_c + 2\alpha}$$

where N_c is the number of documents in class c. The binary feature assumption can sometimes simplify the model by focusing on the presence or absence of terms rather than their frequency, which can be beneficial in highly sparse datasets.

Complement Naive Bayes (ComplementNB)

Complement Naive Bayes (ComplementNB) is a variant designed to handle imbalanced data more effectively. ComplementNB modifies the calculation by focusing on the complement of each class (i.e., treating non-occurrences of words as separate features). This approach aims to improve performance, particularly in imbalanced datasets, where classes may have unequal representation. The smoothing parameter $\alpha = 0.1$ was used here as well to control overfitting.

4.3.3 Voting Classifier (Ensemble Model)

To enhance prediction accuracy and robustness, I combined the above classifiers in an ensemble using a soft-voting strategy. In soft-voting, each classifier's predicted probabilities are averaged, and the class with the highest probability is selected as the ensemble prediction. This approach leverages the strengths of each classifier, allowing the ensemble to mitigate individual model weaknesses.[7]

The ensemble model's output is given by:

$$\hat{y} = \arg\max_c \sum_i w_i \cdot P(c|x, \theta_i)$$

where:

- $P(c|x, \theta_i)$ is the probability of class c for instance x given the model θ_i.
- w_i is the weight assigned to each classifier i; here, each classifier was given equal weight.

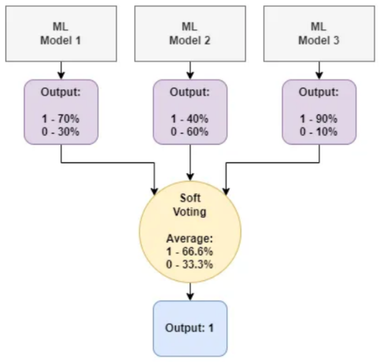

Figure 6. Depiction of Soft Voting in Ensemble Machine Learning

4.3.4 Cross-Validation and Hyperparameter Tuning

I utilized 5-fold cross-validation on the training data to evaluate the ensemble model's generalizability. Cross-validation involves dividing the training set into five subsets, where the model is trained on four subsets and validated on the fifth, repeated five times. This technique provides a robust estimate of the model's performance across different data subsets, reducing the likelihood of overfitting.

$$\text{Accuracy}_{CV} = \frac{1}{K} \sum_{k=1}^{K} \frac{\text{Correct Predictions in Fold}_k}{\text{Total Samples in Fold}_k}$$

where $K = 5$ in this implementation.

Hyperparameter tuning was conducted to further optimize the model. For instance, I varied the smoothing parameter α for each Naive Bayes classifier and the maximum number of iterations (`max_iter`) for the Logistic Regression model. However, these adjustments did not yield significant improvements over the default settings, indicating that the initial parameter choices were effective.

4.4. Presentation of Results

The ensemble model with soft-voting was trained on the IMDB training data and evaluated on the test data. Below is a summary of the evaluation metrics obtained from the model's performance on this sentiment analysis task:

- **Accuracy:** 0.88
- **Precision (Positive Class):** 0.89
- **Precision (Negative Class):** 0.87
- **Recall (Positive Class):** 0.88
- **Recall (Negative Class):** 0.88
- **F1-Score (Positive Class):** 0.88
- **F1-Score (Negative Class):** 0.88

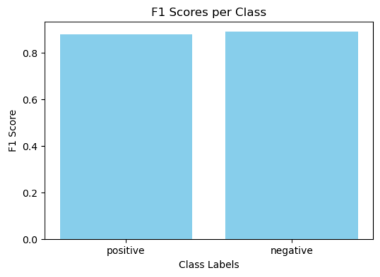

Figure 7. F1-scores per label

The above metrics indicate the model's balanced performance across both positive and negative sentiment classifications. Each metric has been carefully calculated as follows:

- **Accuracy** is the proportion of correctly classified samples across all classes:

$$\text{Accuracy} = \frac{\text{TP} + \text{TN}}{\text{TP} + \text{TN} + \text{FP} + \text{FN}}$$

  where TP, TN, FP, and FN are the counts of true positives, true negatives, false positives, and false negatives, respectively.

- **Precision** measures the accuracy of positive predictions. For the positive class, precision is calculated as:

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

- **Recall** (also known as sensitivity) indicates the model's ability to correctly identify all actual positive instances. For the positive class:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

- **F1-Score** is the harmonic mean of precision and recall, providing a single, balanced metric that is particularly useful when there is an uneven class distribution:

$$\text{F1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}}$$

4.4.1 Confusion Matrix

To further assess the model's performance, I visualized the **Confusion Matrix**. The matrix provided insights into the distribution of predictions across positive and negative classes. The results showed that the model achieved a good balance between positive and negative predictions, with relatively few misclassifications. This indicates that the model is effectively distinguishing between positive and negative sentiments, though some minor misclassification errors still occur.

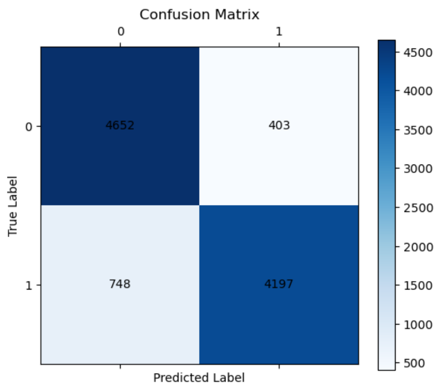

Figure 8. Confusion Matrix

Actual \Predicted	Positive	Negative
Positive	4652	403
Negative	748	4197

4.4.2 Analysis of Results

The evaluation metrics and confusion matrix visualization indicate that the ensemble model with soft-voting performs well in classifying sentiment, achieving an overall accuracy of 88%. The high F1-scores for both positive and negative classes suggest a balanced and reliable performance. However, there is room for improvement, particularly in enhancing precision and recall further.

The current limitations might be due to the feature extraction method, TF-IDF vectorization. Although TF-IDF provides meaningful term-weighting, it may not capture deeper semantic information. A more sophisticated approach, such as incorporating embeddings from pre-trained language models like BERT, could potentially enhance feature representation and improve the model's ability to capture nuances in sentiment.[8]

In future work, I would consider exploring these advanced embedding techniques and tuning hyperparameters further to potentially increase precision and recall in both classes.

5. Conclusion

In this project, I explored the use of Bayesian networks for sentiment analysis in text data, focusing on the IMDB movie review dataset. Bayesian models demonstrated their effectiveness in handling text classification tasks, thanks to their probabilistic nature and ability to generalize well from limited datasets. By integrating MultinomialNB, BernoulliNB, and ComplementNB classifiers with ensemble methods, the project achieved a well-rounded, stable performance in terms of accuracy, precision, and recall. Through careful parameter tuning, such as optimizing the smoothing factor and selecting an n-gram range that captured rich contextual information, the model could better differentiate between positive and negative sentiments.

One notable insight is that even though Bayesian networks are conceptually simpler than many deep learning approaches, they remain effective for text classification tasks. Their probabilistic foundations make Bayesian networks particularly resilient to the challenges of sparse and imbalanced data, which is common in sentiment analysis. This characteristic also means they are computationally efficient and less prone to overfitting on small datasets, making them suitable for real-world applications where computational resources might be limited.

Additionally, this project emphasized the importance of feature extraction techniques like TF-IDF and n-grams, which were instrumental in enhancing the model's ability to understand the context of sentiment words. The n-gram range (1,3) proved especially useful in capturing nuanced expressions of sentiment by including single words, pairs, and triplets, which better reflects the variety and subtlety of human language.

In future work, a deeper exploration could involve testing more advanced natural language processing models, such as transformers or RNNs, and comparing them to Bayesian networks on interpretability and computational efficiency. Despite recent advances in deep learning, this project underlines the importance of classical machine learning methods for tasks where explainability, computational simplicity, and efficiency are critical.

References

[1] Kanwarpartap Singh Gill, Vatsala Anand, Rahul Chauhan, Ankur Choudhary, and Rupesh Gupta. CNN, LSTM, and bi-LSTM based self-attention model classification for user review sentiment analysis. In *2023 3rd International Conference on Smart Generation Computing, Communication and Networking (SMART GENCON)*, pages 1–6. TLDR: This study focuses on the use of convolutional neural networks, long short-term memory (LSTM), and bidirectional LSTM with access to selfattention mechanism for sentiment analysis of user reviews with the highest accuracy rate among the three.

[2] Manjot Kaur and Aakash Mohta. A review of deep learning with recurrent neural network. In *2019 International Conference on Smart Systems and Inventive Technology (ICSSIT)*, pages 460–465. TLDR: Recurrent Neural Network (RNN) is a deep learning model that uses the concept of supervised learning to process examples one at a time, preserving an element, that reflects over a long period of time.

[3] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need.

[4] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

[5] Ajay Shrestha and Ausif Mahmood. Review of deep learning algorithms and architectures. 7:53040–53065. Conference Name: IEEE Access TLDR: This paper reviews several optimization methods to improve the accuracy of the training and to reduce training time, and delve into the math behind training algorithms used in recent deep networks.

[6] Qing Liu, Jing Wang, Dehai Zhang, Yun Yang, and NaiYao Wang. Text features extraction based on TF-IDF associating semantic. In *2018 IEEE 4th International Conference on Computer and Communications (ICCC)*, pages 2338–2343. TLDR: The word2vec model is used to train the word vector in the corpus to obtain its semantic features and constructing a VSM (vector space model) with these clusters as feature units can effectively improve the accuracy of text feature extraction.

[7] Ali Athar, Sikandar Ali, Muhammad Mohsan Sheeraz, Subrata Bhattachariee, and Hee-Cheol Kim. Sentimental analysis of movie reviews using soft voting ensemble-based machine learning. In *2021 Eighth International Conference on Social Network Analysis, Management and Security (SNAMS)*, pages 01–05. TLDR: This paper has proposed a state-of-the-art soft voting ensemble (SVE) approach to perform sentimental analysis of movie reviews, which outperformed all other classifiers by giving an overall accuracy, precision, recall, and f1-score of 89.9%, 90.0%, and 90.1%, respectively.

[8] Zhengjie Gao, Ao Feng, Xinyu Song, and Xi Wu. Target-dependent sentiment classification with BERT. 7:154290–154299. Conference Name: IEEE Access TLDR: Experiments show that the TD-BERT model achieves new state-of-the-art performance, in comparison to traditional feature engineering methods, embedding-based models and earlier applications of BERT, in relation to aspect-based sentiment analysis.