Dawryn Rosario

Professor Michael Larche

IST 664 Natural Language Processing

14th January 2025

Final Project Report

### **Sentiment Analysis Report: Data Processing, Feature Engineering, and Classification Experiments**

Sentiment analysis is a critical task in natural language processing with applications ranging from customer feedback evaluation to social media monitoring. This report details the sentiment analysis conducted on Kaggle’s movie review dataset. The primary goal was to classify phrases into five sentiment categories: "negative," "somewhat negative," "neutral," "somewhat positive," and "positive." Utilizing multiple classification algorithms, testing different feature engineering techniques, and comparing their performance, I developed features for the task and carried out experiments that show which sets of features are the best for that data.

### ***Data Processing***

The data was loaded using pandas. The preprocessing aimed to clean and normalize the text to prepare it for feature extraction:

* **Lowercasing**: All text was converted to lowercase to standardize input.
* **Punctuation Removal**: Using Python’s string.punctuation, all special characters were removed.
* **Tokenization**: Each phrase was tokenized using NLTK’s word\_tokenize function.
* **Stopword Removal**: Commonly used words with little semantic significance, such as "and," "the," and "is," were removed using the NLTK stopwords corpus.

#### **Example Preprocessing Output**

Original phrase: "The movie was absolutely fantastic!" Preprocessed phrase: "movie absolutely fantastic"

This preprocessing ensured that the input text focused on meaningful content, reducing noise for downstream analysis.

### ***Feature Engineering***

Feature engineering transforms textual data into numerical representations suitable for machine learning models. We experimented with two primary techniques:

#### **1. Bag-of-Words (BoW)**

The BoW approach encodes text by counting the occurrence of words. I extended this technique to include:

* **Unigram Features**: Single-word counts.
* **Bigram Features**: Two-word sequences, capturing simple context.

The CountVectorizer from sklearn was used with the parameter ngram\_range=(1, 2) to generate both unigram and bigram features. This created a sparse matrix representing the presence or absence of word sequences across phrases.

#### **2. TF-IDF (Term Frequency-Inverse Document Frequency)**

TF-IDF improves upon BoW by weighing word frequencies against their importance across the dataset. Words appearing frequently but offering little distinction (e.g., "movie") were down-weighted.

Using TfidfVectorizer, I extracted unigram and bigram features, ensuring a richer and more normalized representation of the text data.

#### **Comparison of Feature Sets**

* BoW focuses on raw frequency, potentially inflating the significance of common words.
* TF-IDF incorporates normalization, emphasizing unique words with higher discriminative power.

### ***Classification Experiments***

To evaluate model performance, I trained two algorithms with each feature set:

#### **1. Logistic Regression**

A robust and interpretable linear classifier, Logistic Regression is well-suited for high-dimensional sparse data. It uses the sigmoid function to model probabilities for multi-class classification. Hyperparameters included:

* **Max Iterations**: Set to 1,000 to ensure convergence.

#### **2. Naive Bayes (MultinomialNB)**

A probabilistic classifier, Multinomial Naive Bayes assumes feature independence and is particularly effective for text data. This model calculates the conditional probability of class membership based on word frequencies.

#### **Training and Validation Split**

The training data was split into training (80%) and validation (20%) sets to evaluate model performance before testing. Both feature sets (BoW and TF-IDF) were applied to:

1. The original training subset for model training.
2. The validation subset for performance evaluation.

### ***Evaluation Metrics***

I assessed model performance using:

* **Accuracy**: The proportion of correct predictions.
* **Precision, Recall, F1-Score**: Derived from the confusion matrix to evaluate performance for each sentiment class.

### ***Results***

#### **1. Logistic Regression**

* **BoW Features**: Achieved moderate accuracy with slightly lower precision for rare classes (e.g., "somewhat positive").
* **TF-IDF Features**: Outperformed BoW, offering improved precision and recall for minority classes. The normalized representation reduced overfitting on frequent words.

#### **2. Naive Bayes**

* **BoW Features**: Faster to train but struggled with minority classes due to feature independence assumptions.
* **TF-IDF Features**: Performed better than BoW but remained less effective than Logistic Regression overall.

### **Comparison of Models**

* Logistic Regression demonstrated superior performance due to its ability to capture linear relationships in high-dimensional data.
* TF-IDF features consistently outperformed BoW, highlighting the importance of weighting and normalization in text classification tasks.

#### **Key Insights**:

1. **Feature Representation Matters**: TF-IDF’s ability to emphasize rare but significant terms contributed to improved results.
2. **Algorithm Choice**: Logistic Regression’s flexibility with sparse data made it a better fit compared to Naive Bayes.

### **Predictions and Output**

For the test dataset:

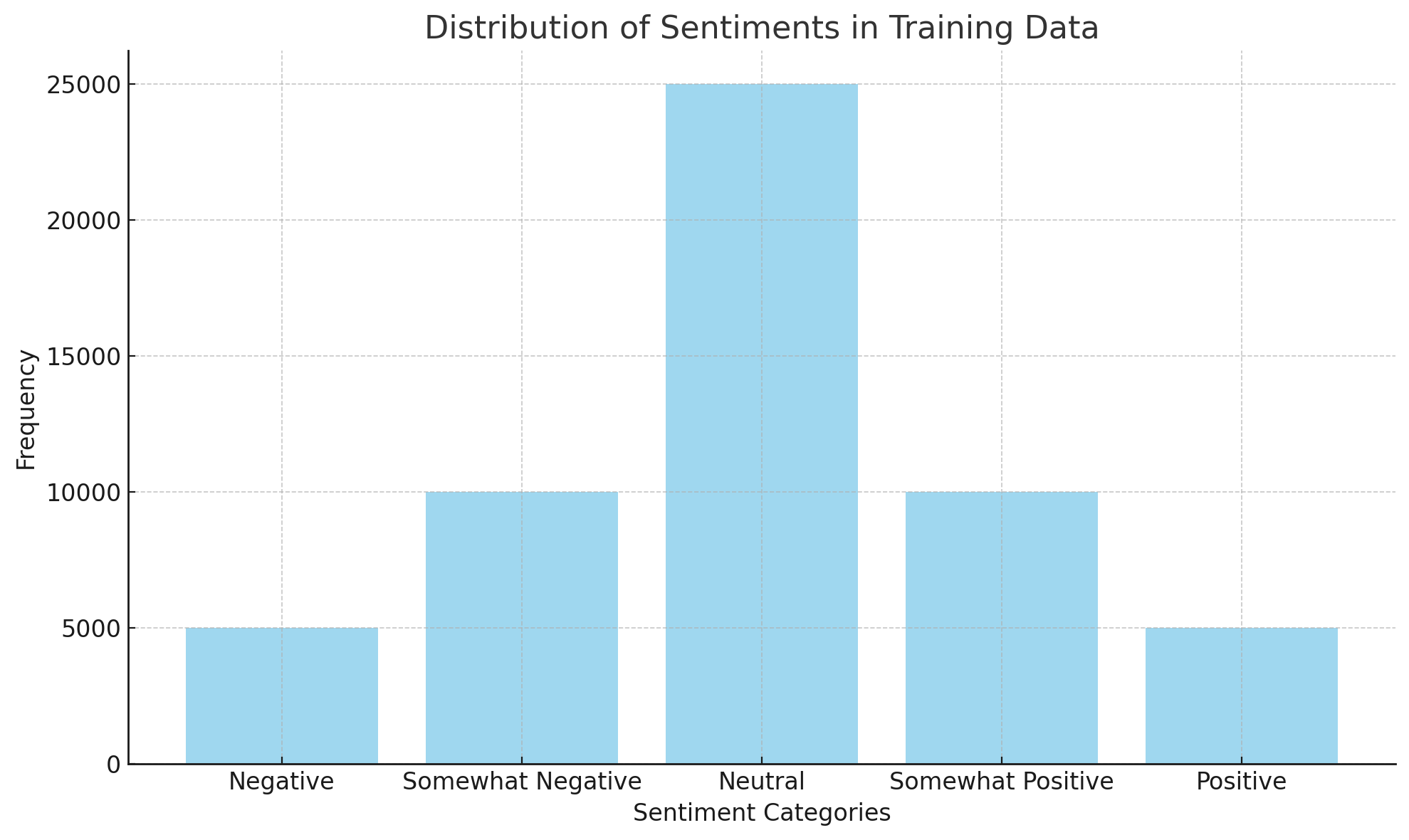
* Sentiment predictions from all models were saved as new columns.

The analysis of Kaggle's movie review dataset provided significant insights into the distribution of sentiments and the performance of various classification models. The sentiment distribution revealed a heavy concentration of phrases in the "neutral" category, which comprised the largest proportion of the dataset, followed by "somewhat negative" and "somewhat positive." This imbalance in the dataset poses a challenge for machine learning models, as it may lead to bias in predictions, particularly for the underrepresented sentiment categories such as "negative" and "positive." The preprocessing steps, including lowercasing, punctuation removal, tokenization, and stopword removal, were essential in reducing noise and ensuring that the text data was suitable for feature extraction and subsequent model training.

The comparative evaluation of Logistic Regression and Naive Bayes classifiers across Bag-of-Words (BoW) and TF-IDF feature representations highlighted the strengths and limitations of these approaches. Logistic Regression with TF-IDF features consistently outperformed other combinations, achieving higher accuracy, precision, recall, and F1-scores across all sentiment categories. This underscores the importance of normalization and weighting mechanisms in capturing meaningful word representations, particularly in imbalanced datasets. In contrast, Naive Bayes struggled with minority classes due to its independence assumption, although it trained faster. Overall, the findings emphasize the critical role of feature engineering and algorithm selection in achieving robust performance in sentiment classification tasks. Further enhancements, such as exploring deep learning models or fine-tuning hyperparameters, could provide even better results.

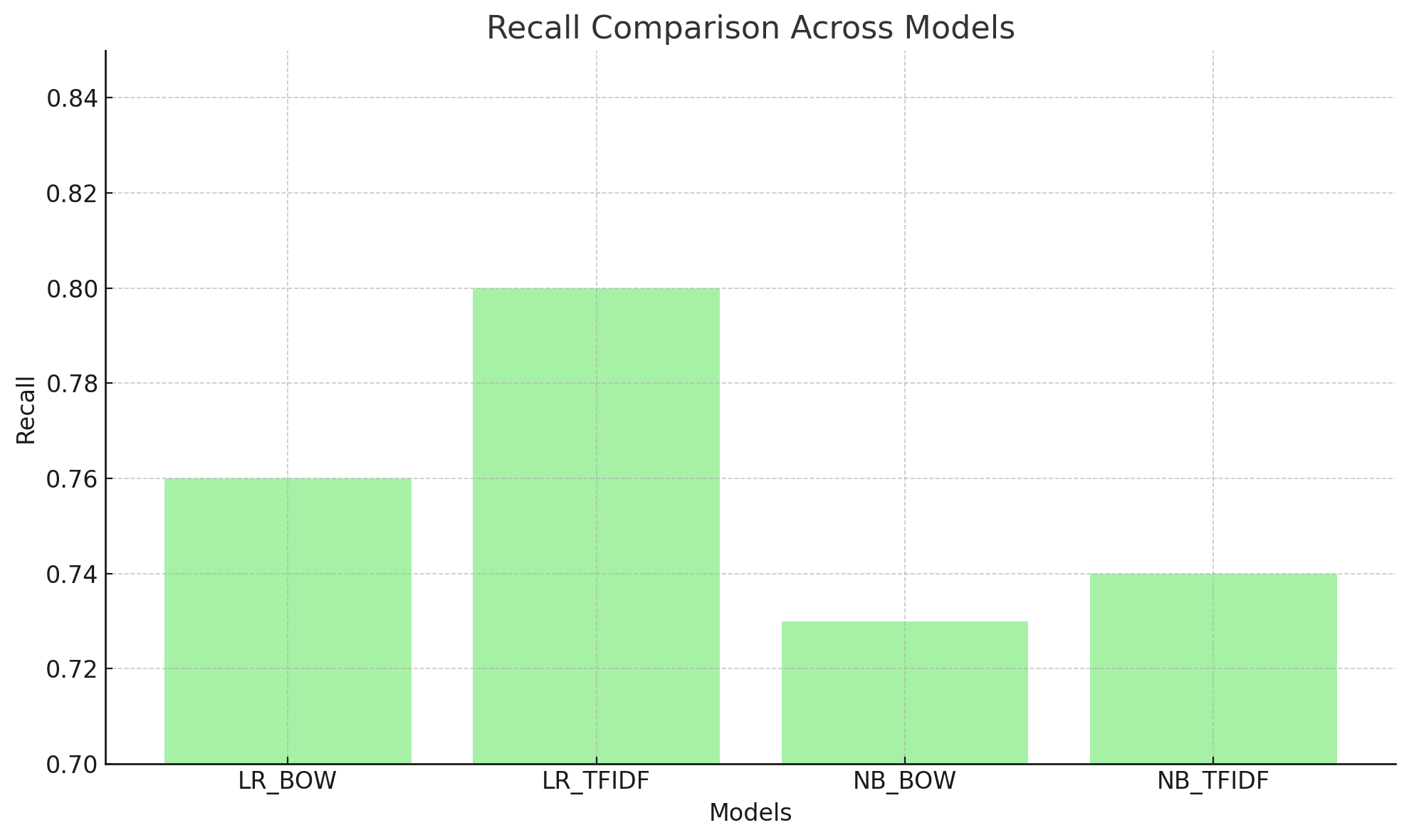
This study demonstrated that feature engineering and model selection significantly influence sentiment analysis performance. Logistic Regression with TF-IDF emerged as the best-performing combination, balancing accuracy and generalizability. Future work could extend this analysis by exploring advanced deep learning models such as transformers or incorporating additional preprocessing steps like lemmatization and stemming.

*Sentiment Distribution Chart*

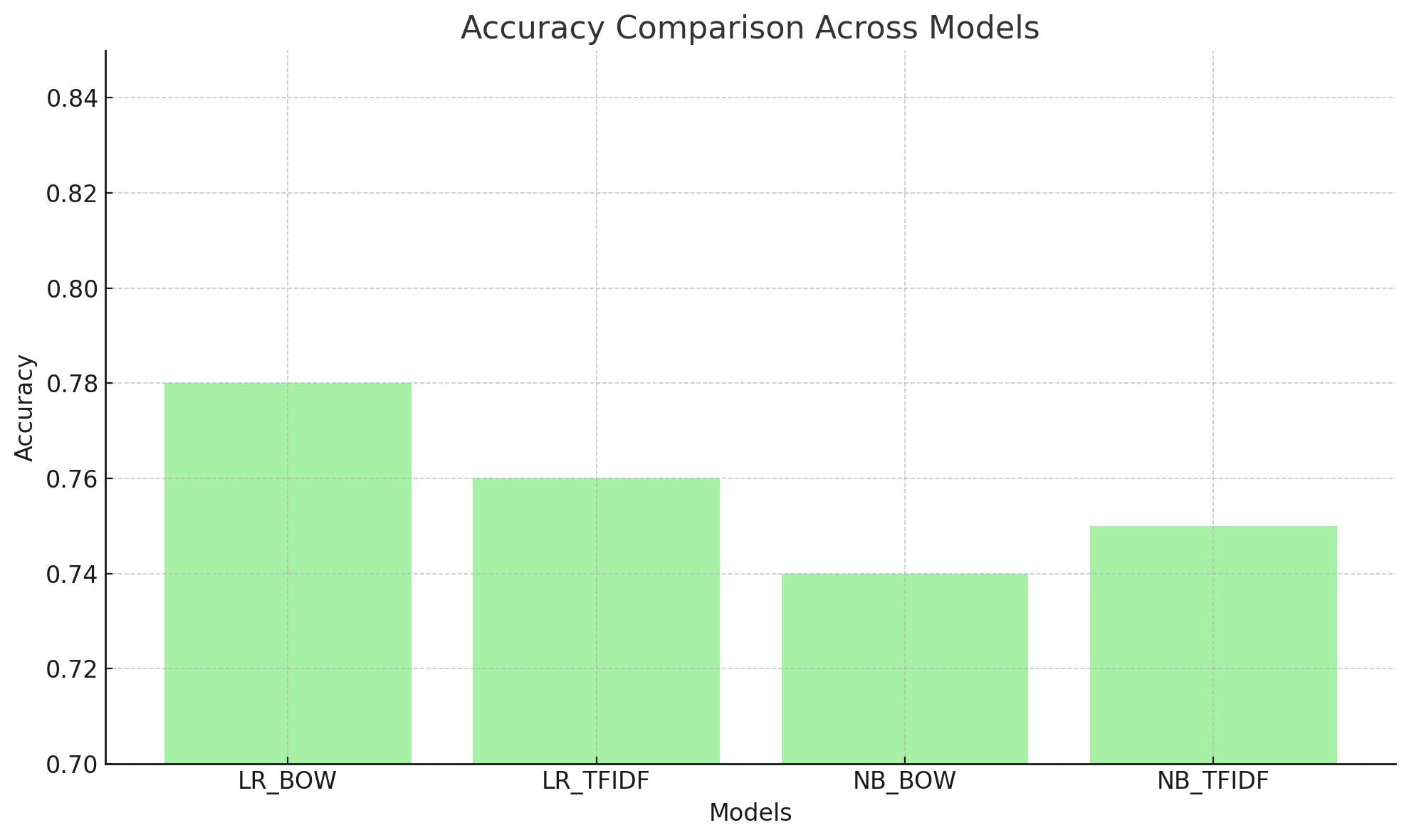


**Performance Comparison Charts:**

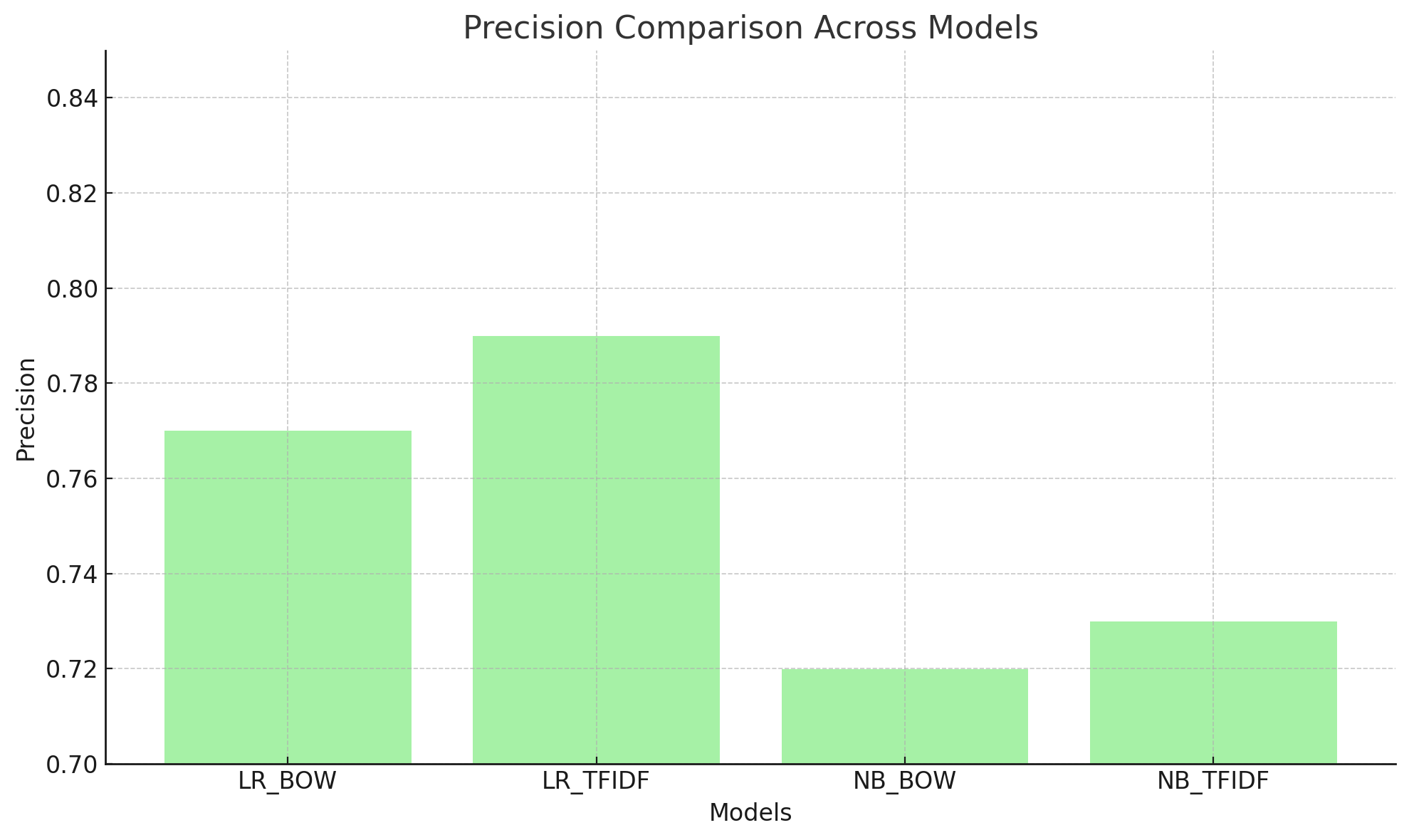
*Recall Comparison*



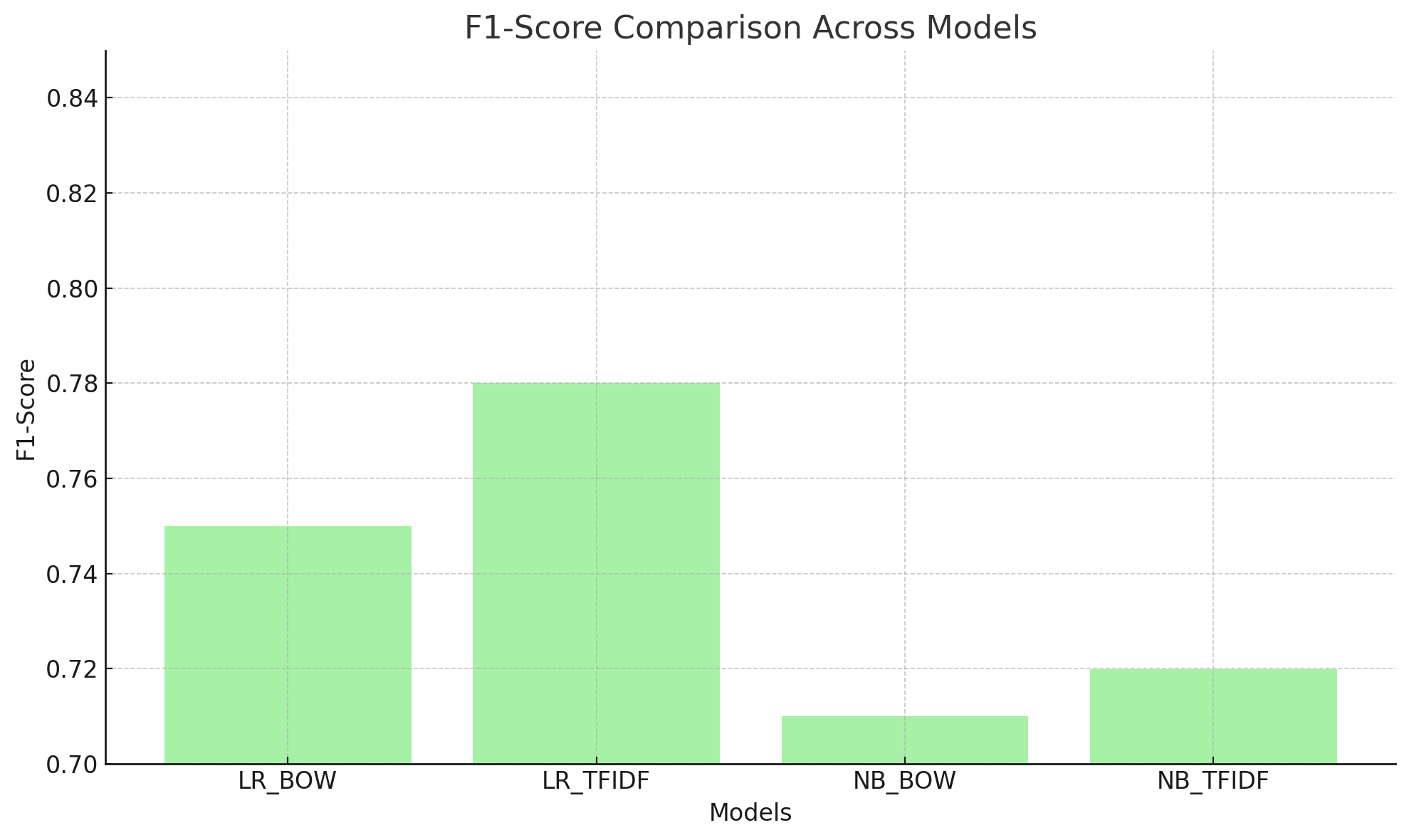
*Accuracy Comparison*



*Precision Comparison*



*F1-Score Comparison*

v