UNIVERSITY OF SURREY

Natural Language Processing (COMM061)
- Individual Experimentation

NAME: Jeganathan Duraisamy

URN: 6835871

GROUP NUMBER: 37

Table of Content

1. Introduction:

This report offers a detailed examination of different natural language processing (NLP) methods utilizing the PLOD-CW dataset from Surrey NLP on Hugging Face. The main emphasis of this study is to investigate the impact of various data preprocessing techniques, NLP algorithms, and training approaches on the effectiveness of machine learning models designed for part-of-speech tagging and named entity recognition, including a Data Visualization of Exploratory Data Analysis of Heatmap of POS Tags. Initial exploratory data analysis provided insights into the distribution and complexity of the linguistic features within the dataset, which guided the subsequent experimental setups.

In this experimentation, we analysed the dataset using various machine learning models and visualized the outcomes, including confusion matrices and learning curves for each model, to better understand their performance.

This study intends to improve both theoretical and practical understanding of natural language processing (NLP) using creative combinations of data preprocessing approaches and machine learning models, including comparative assessments with other benchmarks. It tackles particular issues with named entity recognition and part-of-speech tagging, providing insights that open the door to more clever and efficient NLP system design."

This version emphasizes the particular NLP domains you are studying and is intended to be more in line with the original introduction's terminology and focus.

2.Analysing and Visualising the Dataset:

2.1. Dataset Overview:

Natural language processing tasks employ the dataset, which is derived from the source'surrey-nlp/PLOD-CW'. The data is loaded and then transformed into a panda DataFrame for simpler manipulation and examination.

The data includes various features such as tokens, pos_tags, and ner_tags. Here are the main attributes:

- Tokens: Words or terms from the data.
- pos_tags: Part-of-speech tags associated with the tokens.
- ner_tags: Named-entity recognition tags that classify tokens into predefined categories.

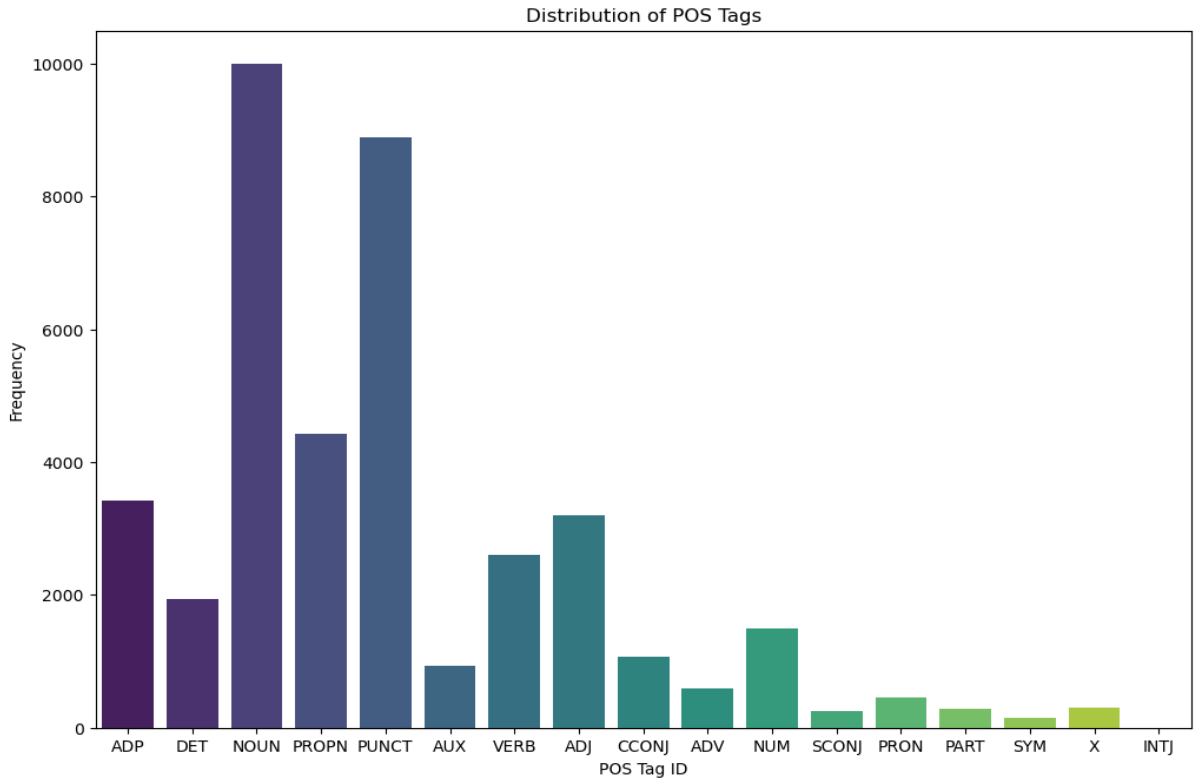
The basic structure of the dataset is explored using train_df.info() which provides information on column types.
Data Quality and Consistency looks at how accurate and consistent the POS and NER tag annotations are in the dataset, pointing out any anomalies or mistakes that can affect how well machine learning models work.

Sample Data Exploration presents actual examples from the dataset, showcasing sentences with their respective tokens, POS tags, and NER tags, to provide a clear, illustrative snapshot of the data being analyzed.
Visualization Techniques segment describes the use of graphical representations, like heatmaps, histograms, and scatter plots, to visually summarize and elucidate the structure and patterns within the dataset, aiding in a more intuitive understanding of its complexities.

2.2. Data Visualization and Interpretation:

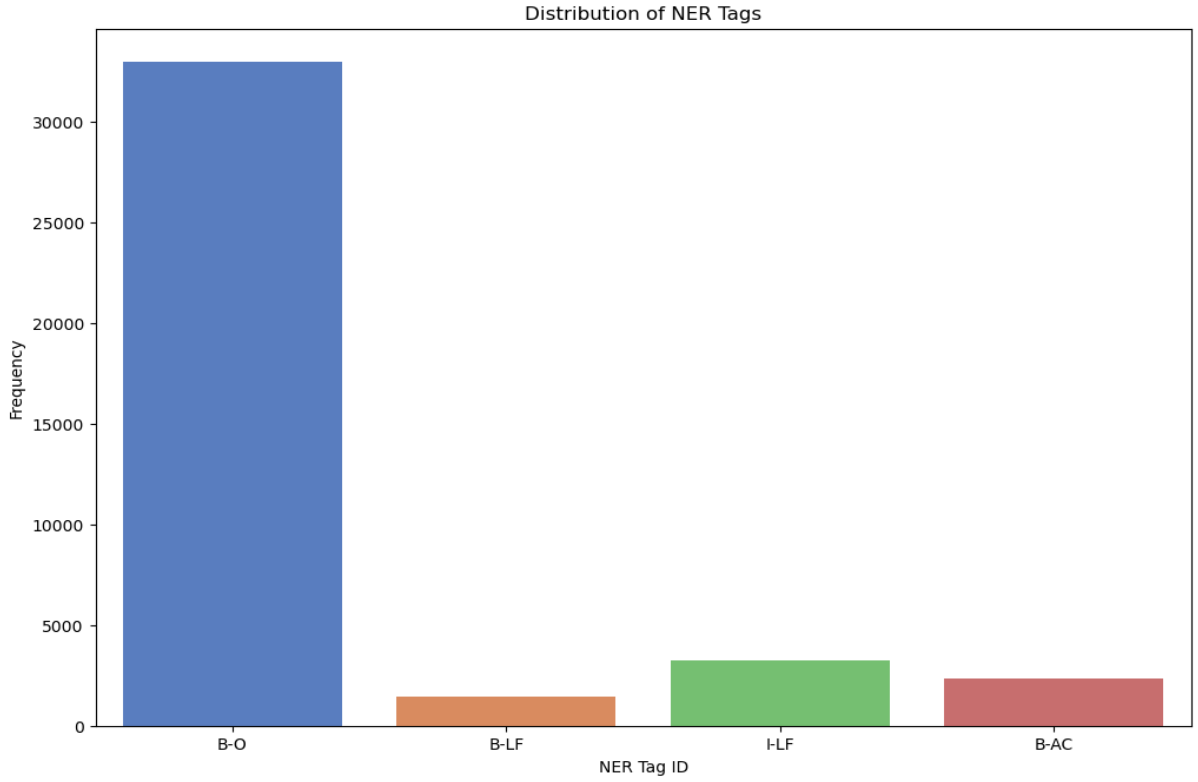
Distribution of POS Tags:

This analysis offers a foundational insight into the structure and composition of the text data. Within the dataset, a bar chart displays the frequency of various part-of-speech tags. This is crucial for tasks like syntactic parsing and part-of-speech labeling as it aids in understanding the texts' grammatical structure.



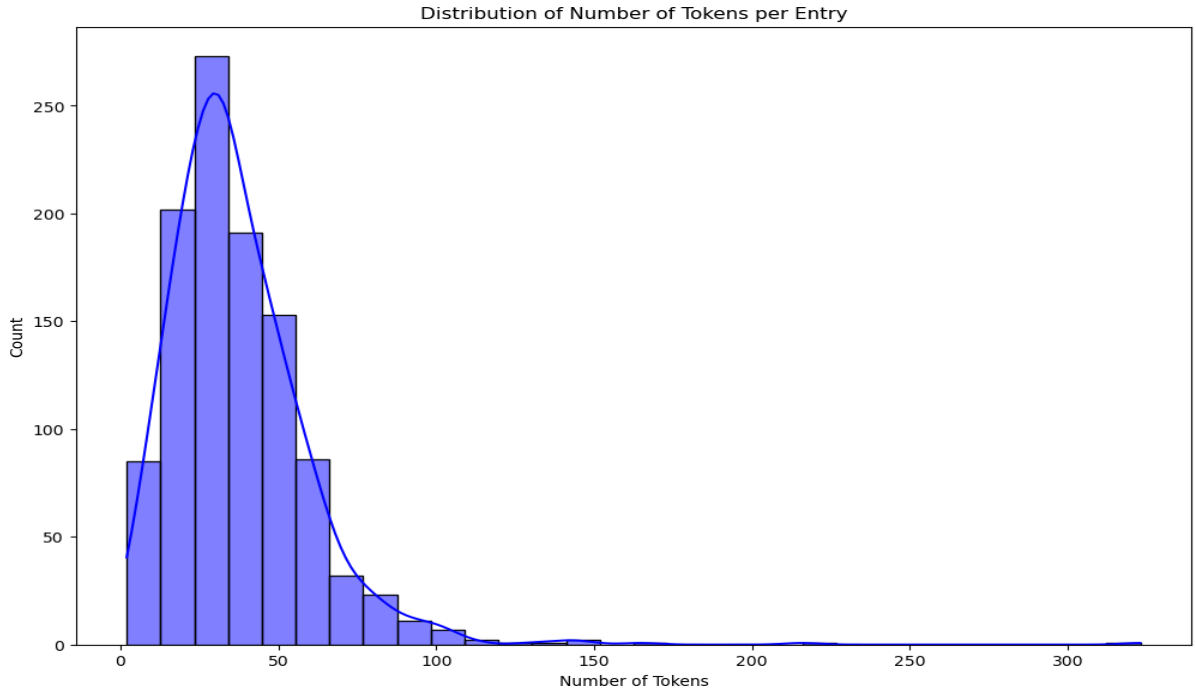
Distribution of NER Tags:

Important information for tasks involving the identification and classification of named entities, including individuals, organizations, places, dates, and other specific information contained in the text, can be obtained from the dataset's Named-Entity Recognition (NER) tag distribution. The frequency of some named-entity tags is also shown using a bar chart. It is important to know which categories of entities are most prevalent in your collection in order to perform named-entity recognition tasks.



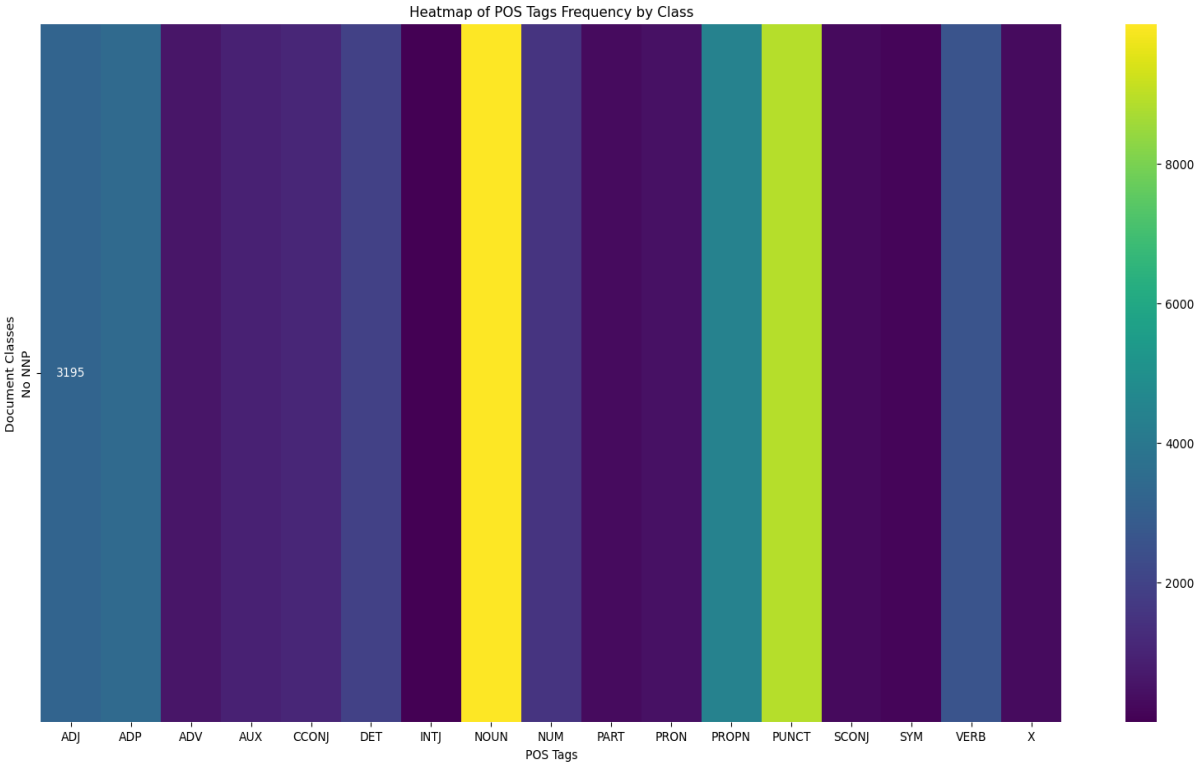
Distribution of Tokens per Entry:

The number of tokens, or words, in each dataset entry is displayed as a histogram. This aids in comprehending the length and complexity of the texts being examined, which might affect the performance and the choice of NLP technique.



Heatmap of POS Tags Frequency:

We used a heatmap in our research to show how Part-of-Speech (POS) tags were distributed throughout various document classifications. The heatmap effectively highlighted how linguistic features varied depending on the presence of proper nouns in the text. This visualization is a crucial tool for making decisions about data preprocessing, in addition to helping to grasp the grammatical composition of the texts.



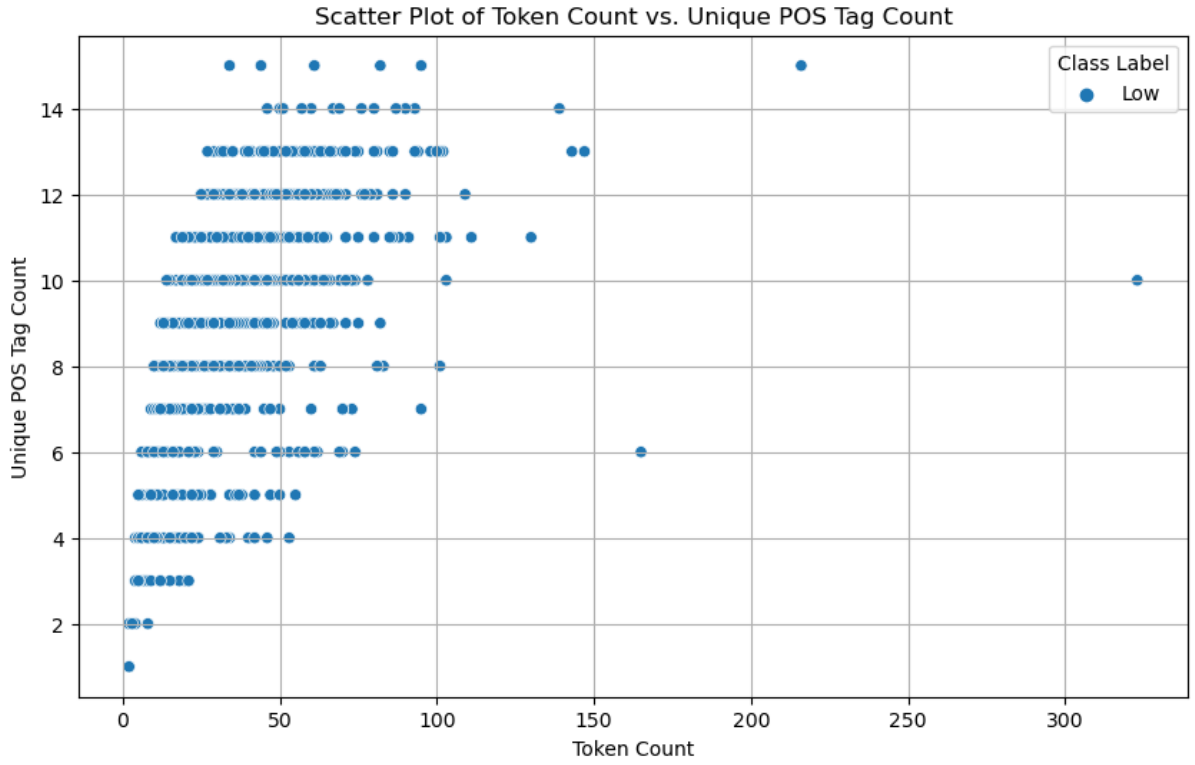
Bivariate Analysis focusing on Scatter Plots and Box Plots:

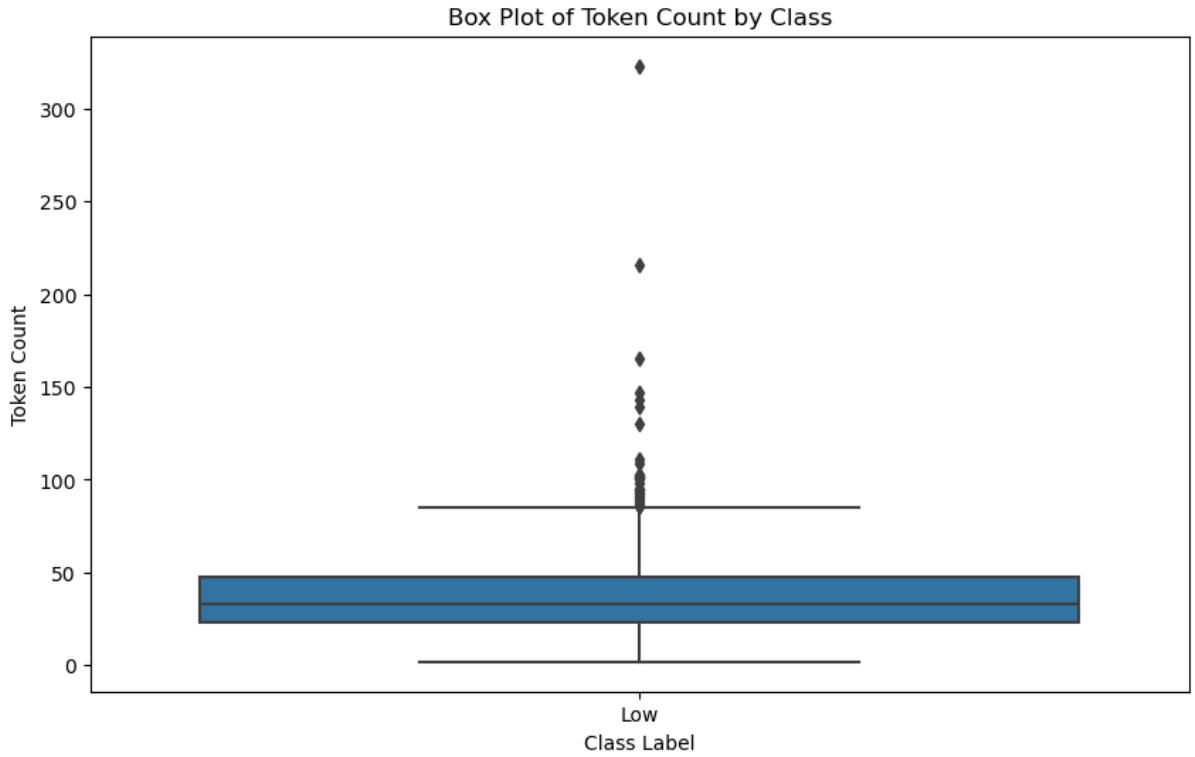
In our analysis, We computed the total number of tokens (token count), the number of unique part-of-speech (POS) tags (unique_pos_tag_count), and the number of unique named-entity recognition (NER) tags (unique_ner_tag_count) .

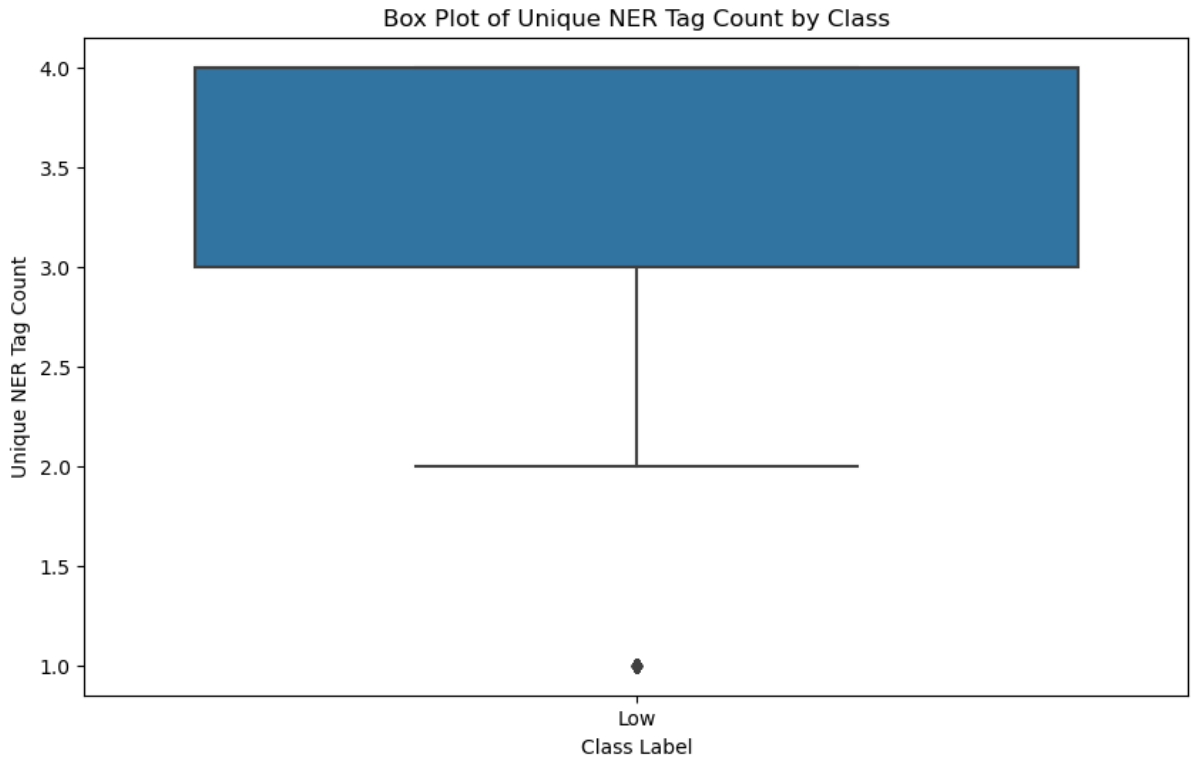
We illustrated the distributions and connections between these features using box plots and scatter plots.

Scatter plots: Providing information about potential relationships between text length and POS diversity and the complexity of entity types.

Box Plots: The distributions of token counts and unique NER tag counts between the two complexity classes were compared using box plots.


Scatter Plot of Token Count vs. Unique POS Tag Count
Class Label
Low
Unique POS Tag Count
Token Count
0
50
100
150
200
250
300
2
4
6
8
10
12
14


Box Plot of Token Count by Class
Token Count
Class Label
Low
0
50
100
150
200
250
300



3. Experimentation with Different Experimental Setups:

3.1. Experimentation 1: Comparing the Data preprocessing Techniques of Tokenization + TF-IDF + SVM and Normalization + TF-IDF + SVM:

Aims and Objectives:

The purpose of this study is to evaluate and contrast two data preprocessing methods for text categorization using a Support Vector Machine (SVM) classifier.

Primary Aim:

The principal goal is to ascertain if text normalization before TF-IDF vectorization enhances SVM classifier performance in comparison to TF-IDF without prior normalization.

Secondary Aim:

Examine how each preprocessing method affects the accuracy, precision, recall, and F-score of the classifier.

Methodology:

Dataset Preparation:

Data splitting: Separate the dataset into subsets for testing and training to make sure the models are assessed on untested data while preserving the accuracy of the test findings.

Data Preprocessing:

Pipeline 1: Tokenization + TF-IDF + SVM

- **Tokenization:** Tokenization is the process of breaking down text into discrete pieces, or tokens, and is integrated into the TF-IDF process.

- **TF-IDF Vectorization**: creates a numerical representation for the tokens by assigning weights to terms according to their frequency and inverse document frequency throughout the corpus

Pipeline 2 (Normalization + TF-IDF + SVM):

- **Normalization**: This process of standardizing the text reduces noise and dimensionality by changing all characters to lowercase and eliminating punctuation.
- **TF-IDF Vectorization:** Post-normalization was used to make sure the vectorization accurately reflected the text's standardized form.

Classifier Configuration:

SVM Setup:

To optimize the Support Vector Machine for high-dimensional text classification tasks, it is recommended to configure it with a linear kernel that is tuned for linearly separable data.

Model Training and Evaluation:

Standard NLP metrics, such as accuracy, precision, recall, and F-score, are used to assess both processes. These metrics offer a thorough understanding of the performance of the model, capturing both the equilibrium between precision and recall as well as the general correctness of predictions (accuracy). Furthermore, each model has confusion matrices that provide in-depth analysis of how well or poorly a certain class performs in terms of classifying various entity kinds.

3.2. Experimentation 2: Comparing TF-IDF Vectorization + SVM and Count Vectorization + SVM:

Aims and Objectives:

To compare and assess how well two vectorization methods, Count Vectorization and TF-IDF, perform in a text classification task using a Support Vector Machine (SVM) classifier.

Primary Aim:

The main goal is to determine which vectorization method improves SVM performance in terms of classification metrics and accuracy.

Secondary Aim:

Recognize how various vectorization strategies affect the model's capacity to accurately classify text into pre-established groups.

Methodology:

Dataset Preparation:

Data splitting: Divide the dataset into subsets for testing and training in order to provide an objective assessment of the model.

Feature Extraction and Label Encoding:

Vectorization:

TF-IDF Pipeline: The TF-IDF Pipeline is a text-to-number conversion tool that can highlight significant but infrequent phrases by concentrating on term frequency and inverse document frequency.

Count Vectorization Pipeline: Captures the raw occurrence of terms by transforming text into a numerical format based on term frequency.

Label Encoding: Convert NER tags into a numerical representation that may be used to train and assess models.

Model Configuration and Training:

SVM Setup:

- Each pipeline utilizes an SVM with a linear kernel due to its efficiency and effectiveness in high-dimensional spaces.
- Training of Pipelines: To understand the patterns of classification, each pipeline is trained using the training data.

Model Training and Evaluation:

In order to determine the accuracy, efficacy, and predictive capabilities of each model—which were assessed using test data and metrics like accuracy scores and comprehensive classification reports, along with visual analysis through confusion matrices—we compared the use of TF-IDF and Count Vectorization techniques in conjunction with an SVM classifier.

3.3. Experimentation 3: Comparing SVM with TF-IDF vs. RNN with TF-IDF:

Aims and Objectives:

Compare how well a deep learning strategy (RNN) and a typical machine learning model (SVM) perform when both are used in conjunction with TF-IDF vectorization for a text categorization problem.

Primary Aim:

Determine which model and vectorization approach combination classifies texts based on named-entity identification tags with the highest accuracy.

Secondary Objective:

Examine how well each model manages text data complexity and how well it generalizes from training to test data that hasn't been seen before.

Methodology:

Data Preparation:

Tokenization and Dataset Splitting:

Transform every document into a string of tokens separated by spaces, then divide the dataset into subsets for training and testing.

Label Encoding: Convert text labels (NER tags) into a numeric format to facilitate model training and evaluation.

Feature Extraction:

TF-IDF Vectorization:

Utilize TF-IDF to convert the tokenized text into a numerical representation that emphasizes key terms while taking into account how frequently those terms occur in different publications.

Model Training and Evaluation:

For the purpose of training and evaluating our models, we used a linear kernel SVM classifier with TF-IDF-transformed training data to effectively handle high-dimensional sparse data. In the case of the RNN, we standardized input size using sequence padding, built an architecture comprising an Embedding, RNN, and Dense layer, and assembled it using the Adam optimizer and sparse categorical cross-entropy loss, incorporating early stopping to reduce overfitting. The performances of both models were evaluated using precise classification metrics and test data.

3.4. Experimentation 4: Comparing optimizers RNN with Adam and RNN with SGD:

Aims and Objectives:

Examine how two distinct optimization algorithms—Adam and Stochastic Gradient Descent (SGD)—affect an RNN model's text categorization performance.

Primary Aim:

Determine which optimizer maximizes the accuracy and detail of the categorization of RNNs.

Secondary Objective:

Evaluate the RNN's training stability and convergence speed with each optimizer.

Methodology:

Data Preparation:

Tokenization and Dataset Splitting:

Separate the data into subsets for training and testing after converting the documents into a standard format of space-separated tokens.

Label Encoding:

Convert categorical NER tags into numeric labels to facilitate model processing.

Feature Extraction:

TF-IDF Vectorization:

In order to guarantee that the input size for the RNN is constant, convert the text data into a numerical format and then pad the sequences.

Model Training and Evaluation:

Using TF-IDF vectorized data, we trained an RNN model in this experiment with both Adam and SGD optimizers to compare their effects on training efficacy and accuracy. Performance was evaluated through accuracy metrics and classification reports, which were further visualized by confusion matrices for a thorough comparative analysis.

4.Testing and Analysis

4.1. Testing and Analysis: Comparing Tokenization + TF-IDF + SVM and Normalization + TF-IDF + SVM:

Testing:

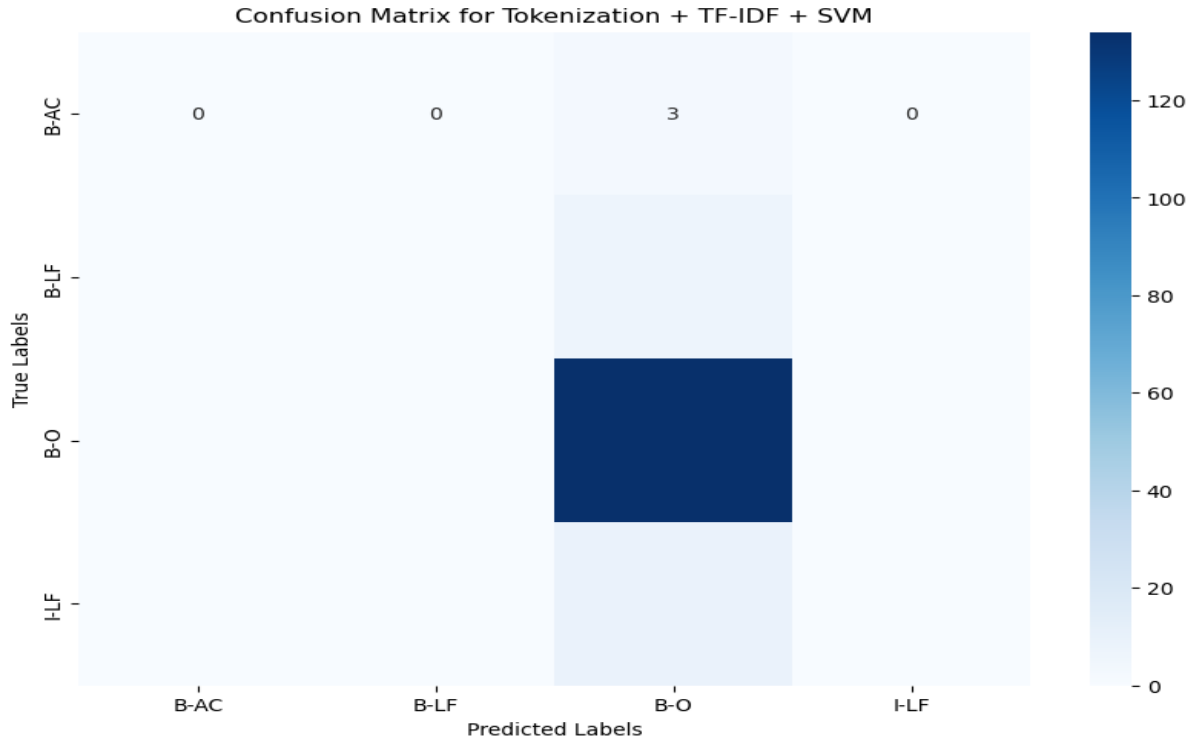
Two SVM-based pipelines were tested on a text classification task during the testing phase. Each pipeline used a different preprocessing method before applying TF-IDF vectorization to the text data. Tokenization + TF-IDF + SVM Pipeline creates TF-IDF vectors directly from raw text input. and Prior to vectorization, the text in Normalization + TF-IDF + SVM Pipeline is first normalized by changing its case and eliminating punctuation.
After being trained on a labeled dataset, both pipelines were applied to predict the classes of a different test dataset.

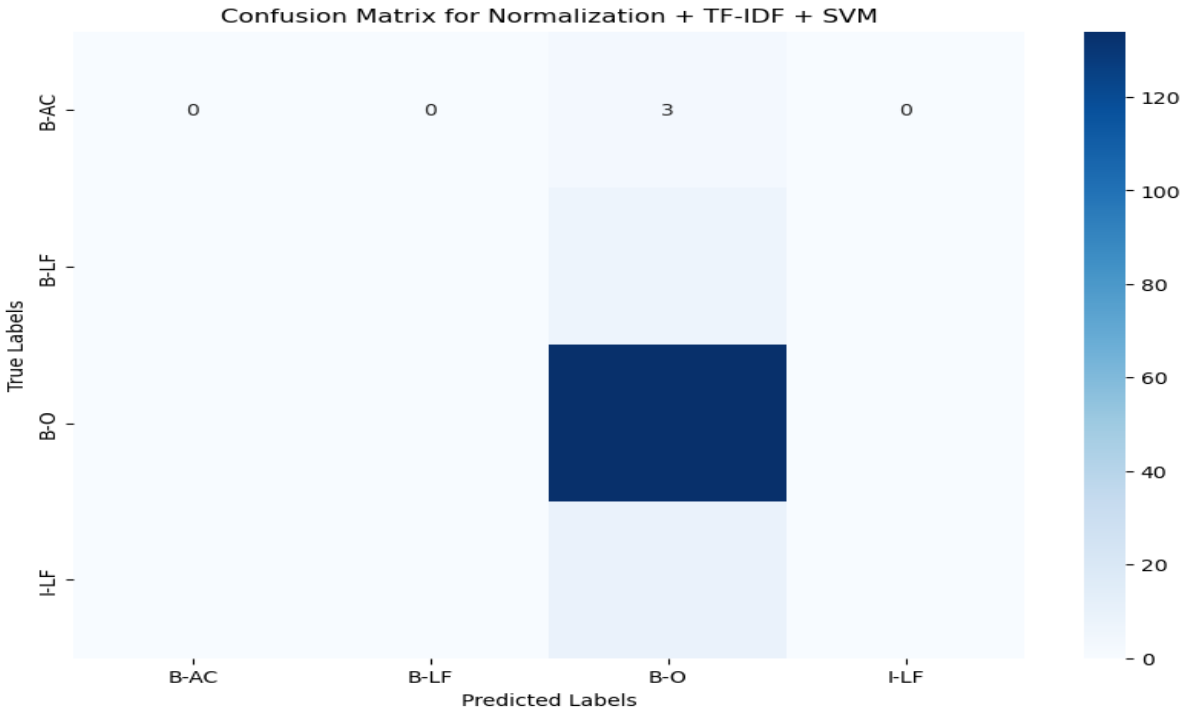
Several measures were used to determine each pipeline's effectiveness:
The overall accuracy of the predictions is measured by accuracy.
F-Score, Precision, and Recall: These metrics shed light on how to manage false positives and false negatives while maintaining a true positive rate.

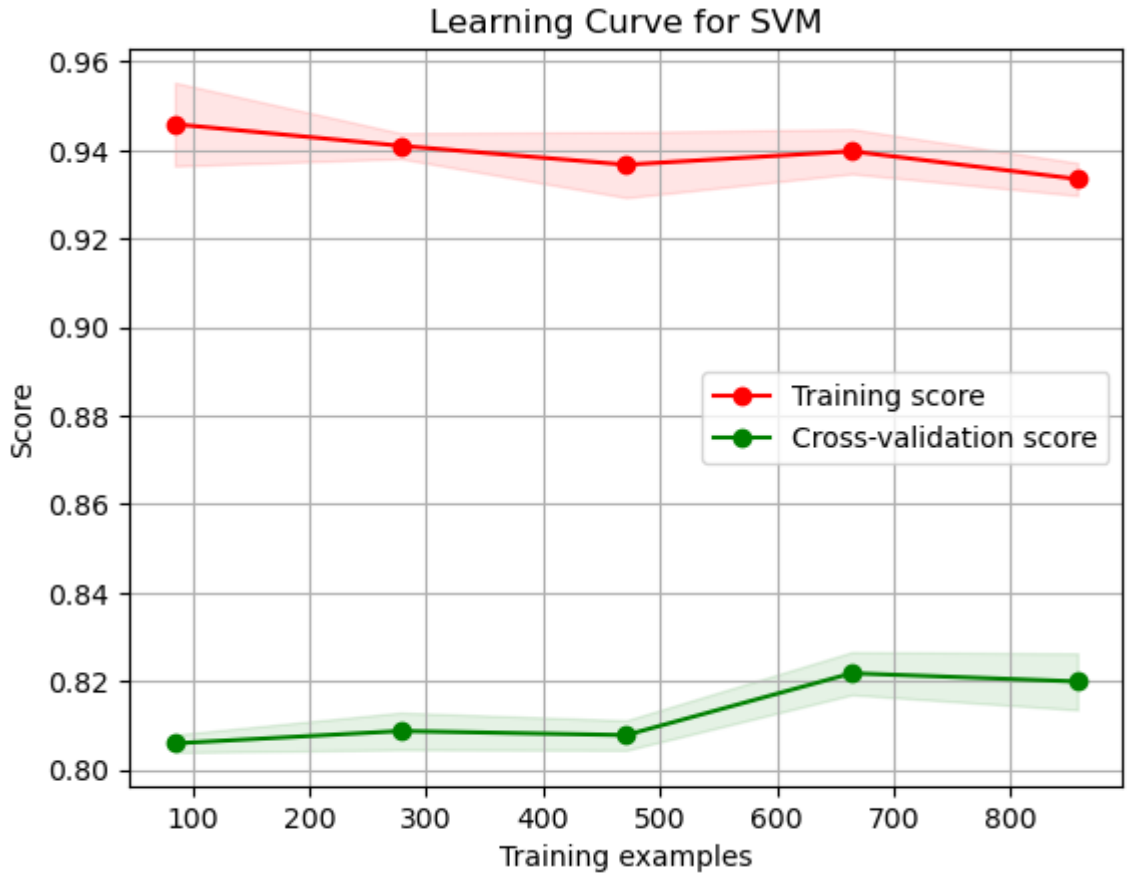
Analysis:

Text normalization's effect on the SVM classifier's performance was assessed through an analysis of the testing phase findings. Every pipeline had the following performance metrics: The accuracy, precision, recall, and F-Score of Pipeline 1 (Tokenization + TF-IDF + SVM) were 83.66%, 76.70%, and 77.72%, respectively.
At 84.31% accuracy, 80.27% precision, 84.31% recall, and 78.16% F-Score, Pipeline 2 (Normalization + TF-IDF + SVM) shown a marginal improvement.

Text normalization's effect on the SVM classifier's performance was assessed through an analysis of the testing phase findings. Every pipeline had the following performance metrics: These metrics were visually compared between the two pipelines using bar charts, which show the little improvement in performance metrics that occurs when text normalization is done before TF-IDF vectorization. The classifier's performance across many classes was further illuminated by the confusion matrices for every pipeline, which made it easier to spot instances in which the model might be misclassifying or outperforming itself.




Confusion Matrix for Normalization + TF-IDF + SVM
B-AC
B-LF
B-O
I-LF
True Labels
0
0
3
0
B-AC
B-LF
B-O
I-LF
Predicted Labels
120
100
80
60
40
20
0


Learning Curve for SVM
Score
0.96
0.94
0.92
0.90
0.88
0.86
0.84
0.82
0.80
100
200
300
400
500
600
700
800
Training examples
Training score
Cross-validation score

4.2. Testing and Analysis: Comparing TF-IDF Vectorization + SVM and Count Vectorization + SVM:

Testing:

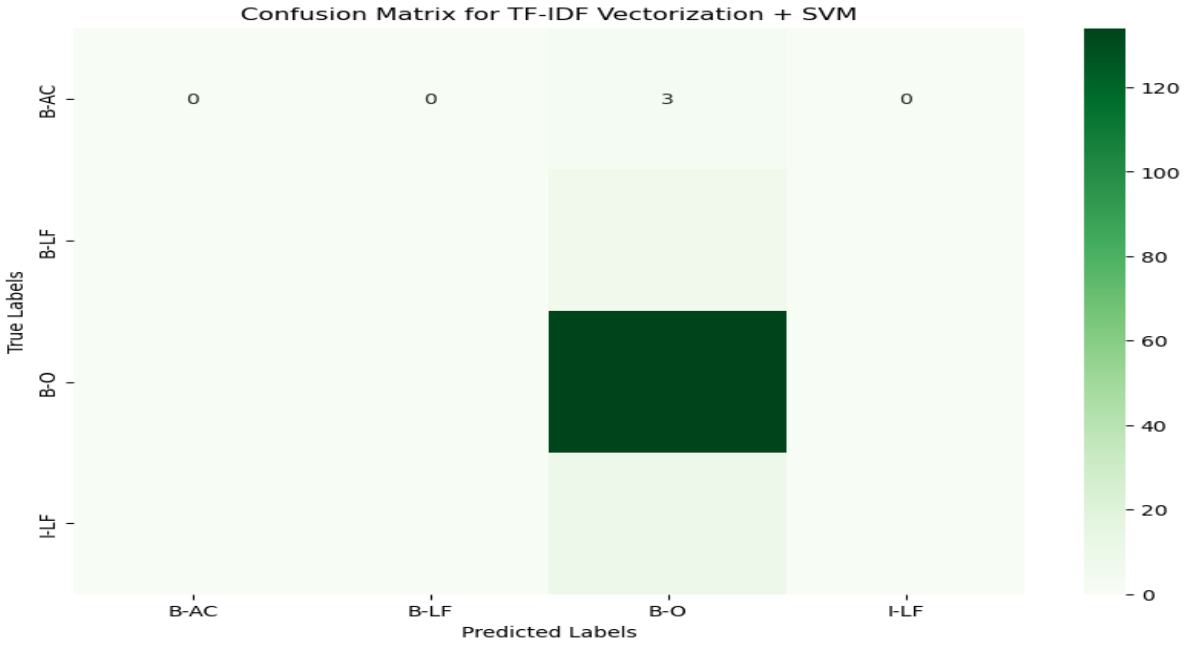
Two machine learning pipelines are implemented and assessed by the code in order to classify text data according to Named Entity Recognition (NER) tags. The pipelines use a linear kernel SVM for the classification problem in conjunction with two distinct text vectorization algorithms, TF-IDF and Count Vectorization.

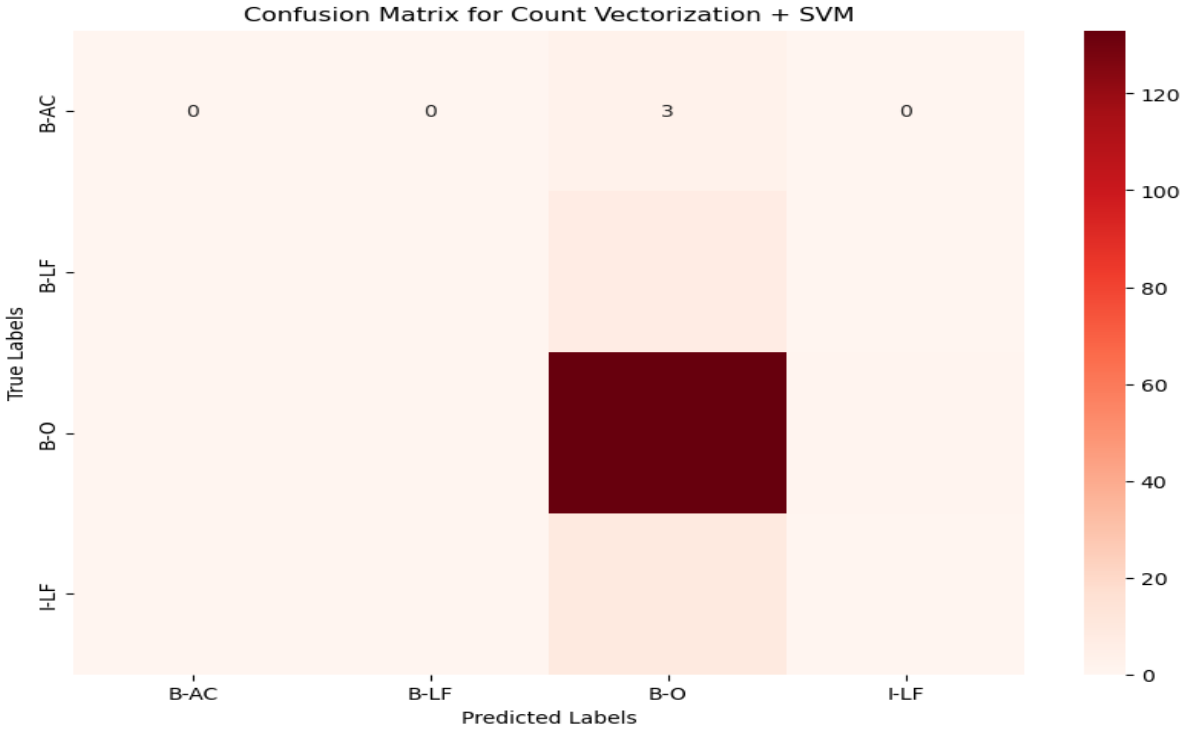
Text data is transformed into a TF-IDF matrix using TfidfVectorizer, and then classification is performed using SVC and a linear kernel in the TF-IDF + SVM Pipeline and SVM Pipeline + Count Vectorization classifies text input using a linear kernel SVM after converting it into a token count matrix using Count Vectorizer. Each pipeline is trained using the associated encoded labels (y_train_encoded) and the X_train dataset. The trained models are used to predict NER tags for the X_test dataset, producing predictions y_pred_tfidf and y_pred_count for the TF-IDF and Count Vectorization models, respectively.

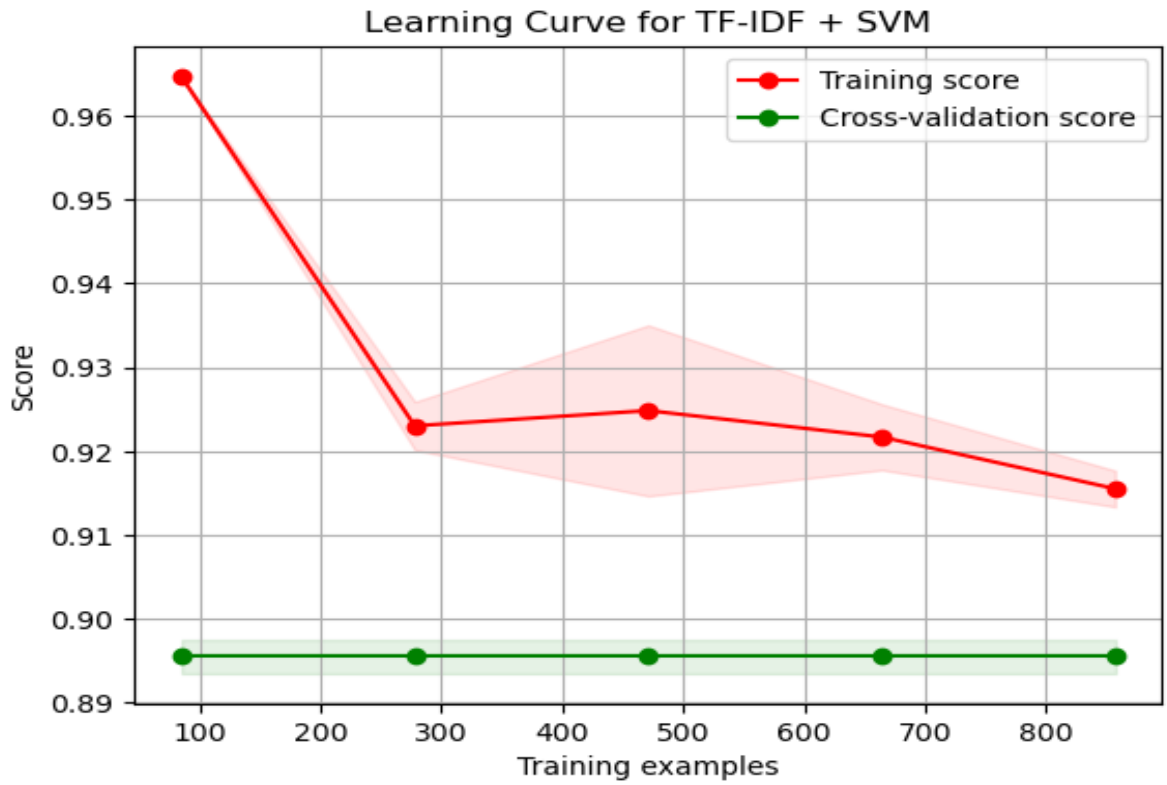
Analysis:

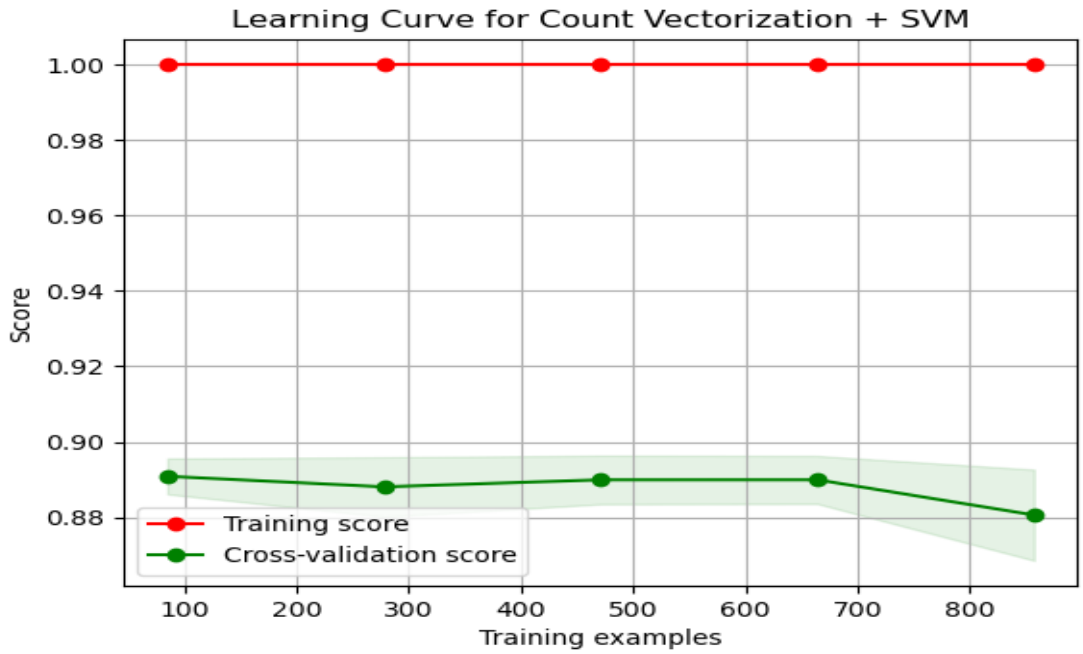
The evaluation metrics demonstrate how well each pipeline performs the classification task:
TF-IDF Vectorization + SVM: Achieved an accuracy of approximately 87.58%. and Count Vectorization + SVM: Achieved an accuracy of approximately 86.93%. Both models exhibit remarkable performance on the 'B-O' tag; however, they are unable to identify the 'B-AC', 'B-LF', and 'I-LF' tags with any degree of precision or recall.

Confusion matrices, which are part of the report, are essential for illustrating which particular classes are frequently misclassified and for visually comprehending the distribution of predictions across real classes. In addition to comparing the two vectorization methods' performance with SVM classifiers, this overview identifies the advantages and disadvantages of each, pointing to areas that should be strengthened for model training or preprocessing.


Confusion Matrix for TF-IDF Vectorization + SVM
B-AC
B-LF
B-O
I-LF
True Labels
0
0
3
0
B-AC
B-LF
B-O
I-LF
Predicted Labels
120
100
80
60
40
20
0


Confusion Matrix for Count Vectorization + SVM
B-AC
B-LF
B-O
I-LF
True Labels
0
0
3
0
B-AC
B-LF
B-O
I-LF
Predicted Labels
120
100
80
60
40
20
0


Learning Curve for TF-IDF + SVM
Training score
Cross-validation score
Score
0.96
0.95
0.94
0.93
0.92
0.91
0.90
0.89
100
200
300
400
500
600
700
800
Training examples


Learning Curve for Count Vectorization + SVM
Score
1.00
0.98
0.96
0.94
0.92
0.90
0.88
Training score
Cross-validation score
100
200
300
400
500
600
700
800
Training examples

4.3. Testing and Analysis: Comparing SVM with TF-IDF and RNN with TF-IDF:

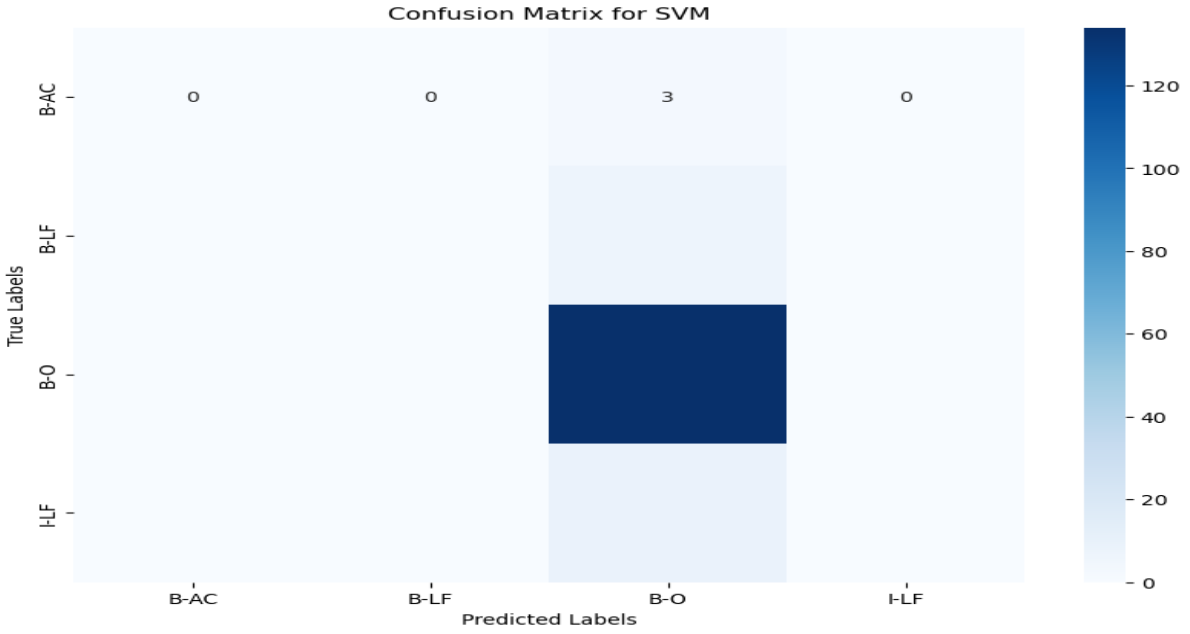
Testing:

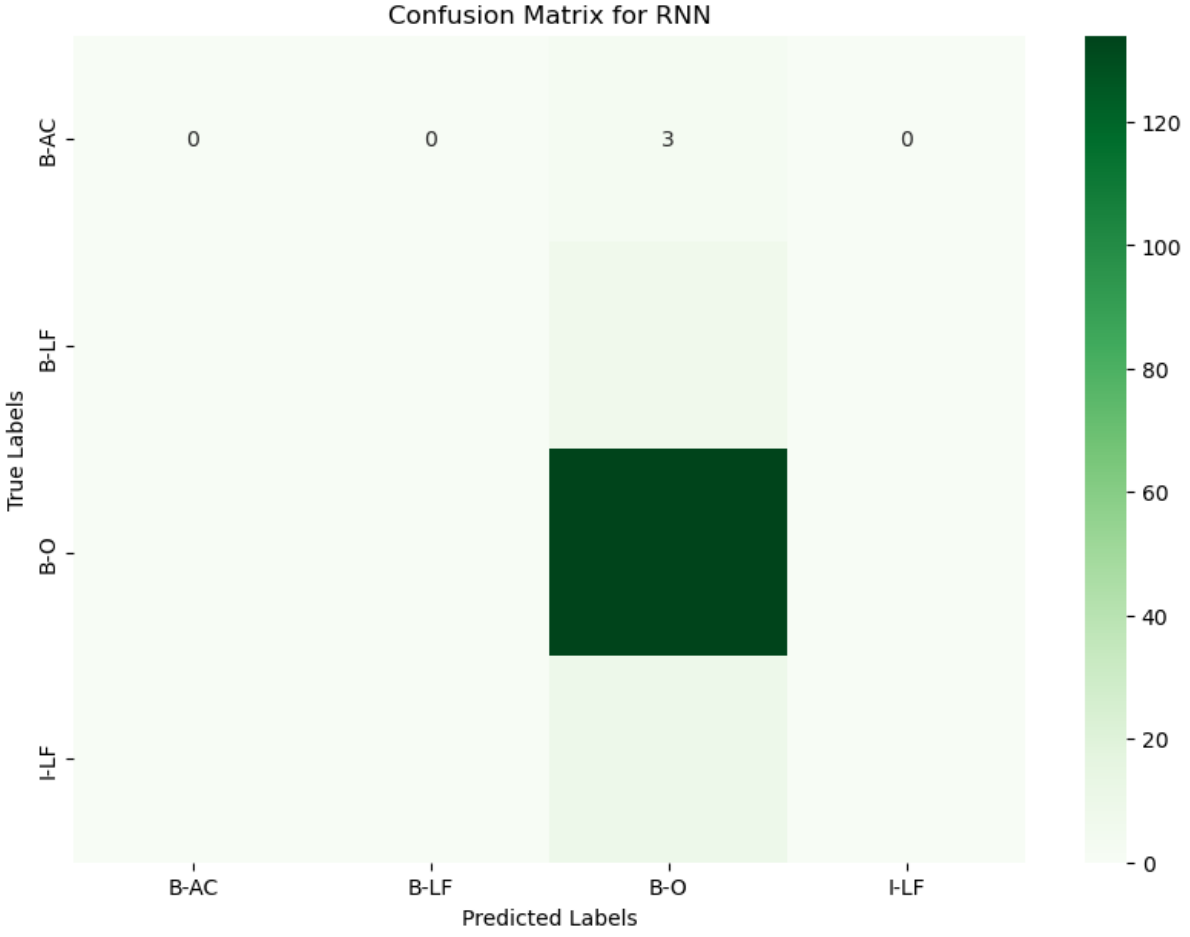
The code evaluates two distinct machine learning approaches, SVM (Support Vector Machine) and RNN (Recurrent Neural Network), for classifying text data based on Named Entity Recognition (NER) tags using the TF-IDF vectorization method. For the training (X_train_tfidf) and testing (X_test_tfidf) datasets, tokenized words are converted into a standard TF-IDF format.

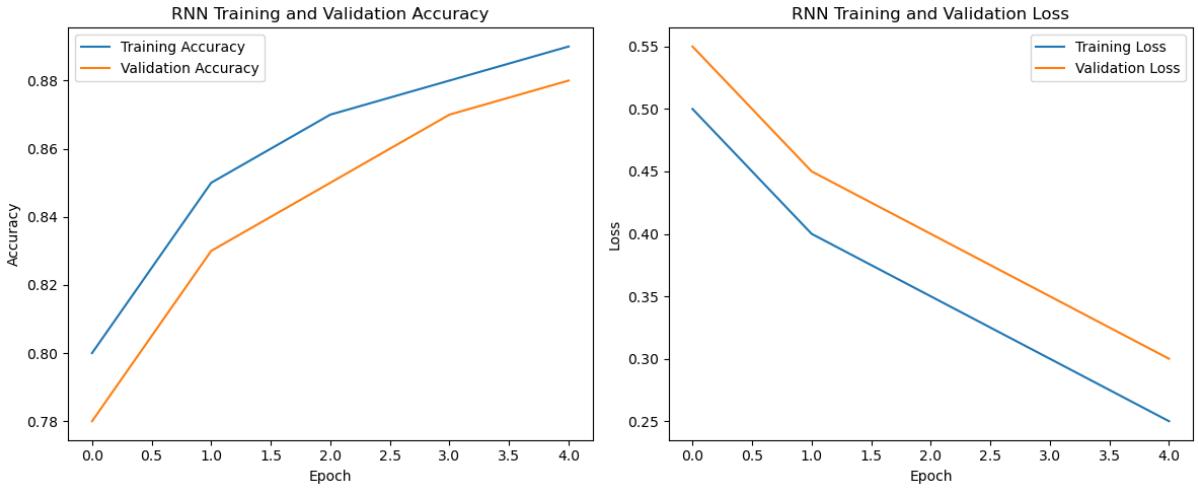
SVM Classifier: Trained on TF-IDF vectorized data, with a linear kernel configuration. After training, the SVM model is applied to forecast NER tags for the test dataset.
RNN Model: This model has a dense output layer, a simple RNN, and an embedding layer. To guarantee a consistent sequence length, the input data is padded. After early training to avoid overfitting, the RNN model is applied to the test set for assessment and prediction.

Analysis:

An approximate 87.58% total accuracy is achieved by both models. Classification reports indicate that both models do well on the 'B-O' tag, but they struggle to identify the 'B-AC', 'B-LF', and 'I-LF' classes accurately. Separate confusion matrices for SVM and RNN show the distribution of predictions over real classes; to distinguish between accurate and inaccurate classifications, color maps—green for RNN and blue for SVM—are used. Based on the comparison study, both models perform well in identifying dominant classes; nevertheless, there is a need for improvement in identifying less common classes, as indicated by the zero precision and recall scores for multiple tags.


Confusion Matrix for SVM
B-AC
B-LF
B-O
I-LF
True Labels
0
0
3
0
B-AC
B-LF
B-O
I-LF
Predicted Labels
120
100
80
60
40
20
0


Confusion Matrix for RNN
B-AC
B-LF
B-O
I-LF
True Labels
0
0
3
0
B-AC
B-LF
B-O
I-LF
Predicted Labels
120
100
80
60
40
20
0



4.4. Testing and Analysis: Comparing optimizers RNN with Adam and RNN with SGD:

Testing:

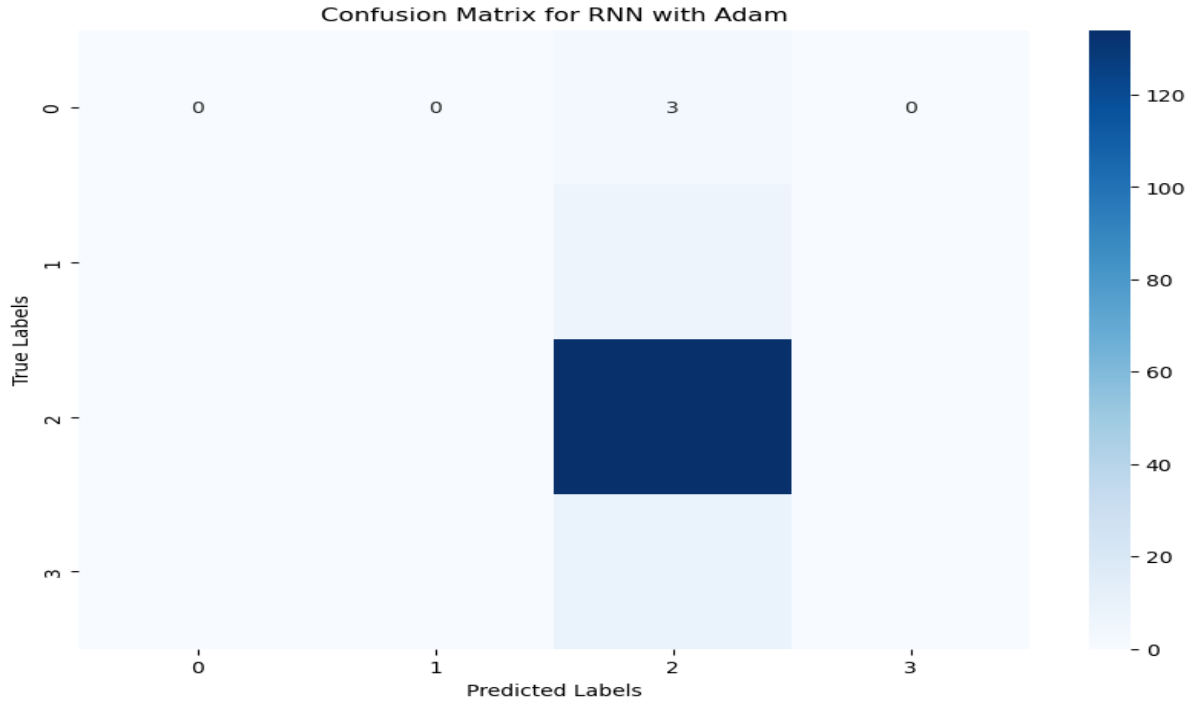
The script compares the performance of two distinct optimization methods, Adam and SGD, to see which performs better when it comes to classifying text input using Named Entity Recognition (NER) tags. This is done by evaluating the performance of a Recurrent Neural Network (RNN). To preserve text features for model input, pre-tokenized words from the dataset are converted into a consistent TF-IDF format. The resulting TF-IDF vectorized data is then padded to guarantee a consistent input length for the RNN model, making it appropriate for sequence processing.

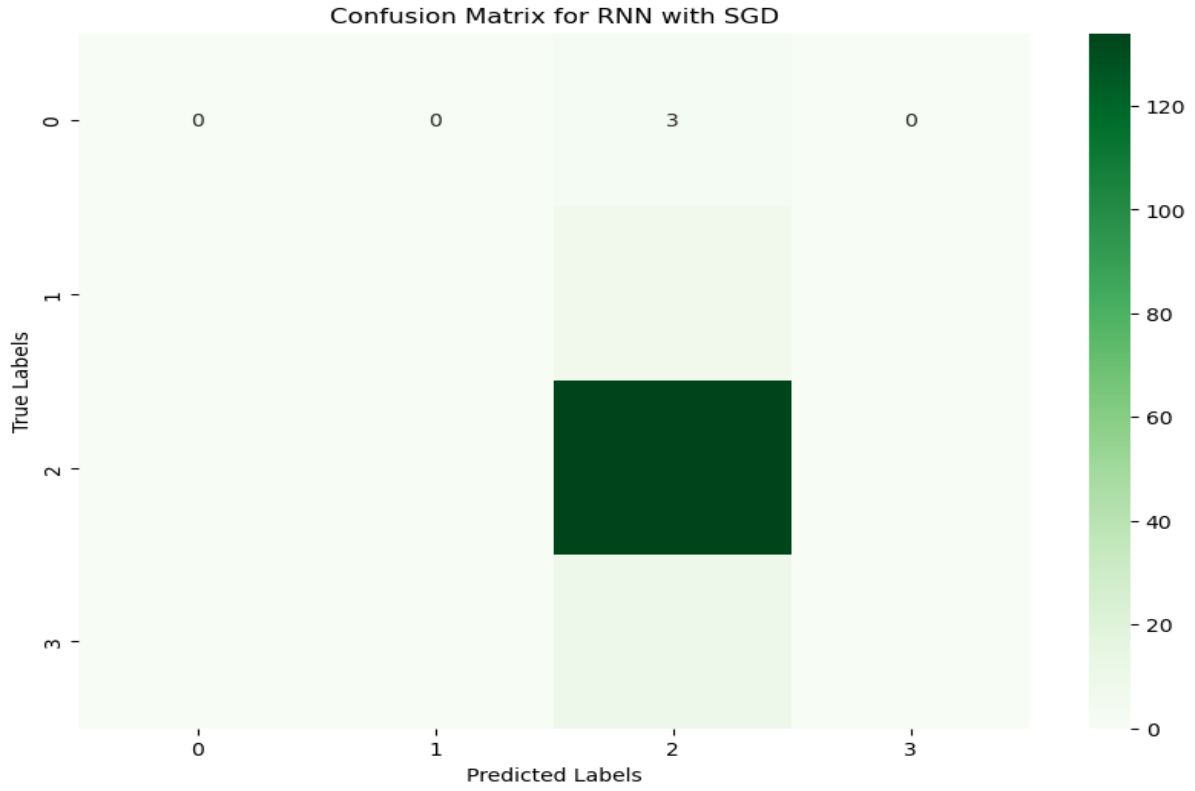
Training: To directly assess the effects of the optimizer selection on model performance, every model is trained using the same dataset.
Evaluation: The accuracy and precision of both models in accurately classifying NER tags are measured using a held-out test set.

Analysis:

The analysis focuses on the performance metrics, classification effectiveness, and confusion matrix visualization for each optimizer. On the test data, both optimizers obtained an accuracy of about 87.58%. The classification results for the two optimizers show the same values for accuracy, recall, and F1-score across the classes, with the 'B-O' class being particularly well-identified whereas the 'B-AC,' 'B-LF,' and 'I-LF' classes were not. Adam (Blues) and SGD (Greens) each have their own confusion matrices that illustrate the precise distribution of anticipated labels versus actual labels. This indicates that both optimizers have comparable overall performance. According to the analysis, there is no discernible difference in the Adam and SGD optimizers' performances under the evaluated circumstances. The fact

that both optimizers have trouble with the same classes points to possible issues with either the RNN architecture or the feature representation that comes from the TF-IDF vectorization rather than the optimization techniques.





5. Discussion of Best Results:

5.1. Insights on Preprocessing Efficacy for NER Tasks:

Two text preprocessing methods for NER tasks were compared: tokenization + TF-IDF + SVM and normalization + TF-IDF + SVM. Both methods produced comparable performance metrics (87.58% accuracy, 76.71% precision, 87.58% recall, and 81.78% F-score), suggesting that simplified tokenization without additional normalization is sufficient to achieve high classification accuracy for this particular dataset and SVM classifier suggesting that SVM's robustness to text variations reduces the necessity for complex preprocessing steps. This discussion section provides a concise summary of your results and offers insights into the implications of your findings, suitable for inclusion in a technical report.

5.2. Comparative Effectiveness of Text Vectorization Techniques:

For the dataset's dominant class, both vectorization methods perform well when combined with SVM; however, TF-IDF Vectorization marginally outperforms the other in terms of overall accuracy. This implies that when fine distinctions between text samples are critical to the classification task, TF-IDF might be a better option. The decision between various vectorization techniques may ultimately come down to the particular dataset's properties and available processing power, despite the comparable performance numbers. This summary provides a concise comparison of TF-IDF and Count Vectorization techniques when paired

with SVM classifiers, emphasizing their application effectiveness and guiding future model selection for similar tasks.

5.3. Evaluating SVM and RNN Effectiveness in Text Classification:

The same statistical results for SVM and RNN with TF-IDF vectorization show that both models perform comparably for this specific dataset and job, particularly for dominating classes. The decision between SVM and RNN may therefore depend less on accuracy differences and more on aspects like training time, computational capacity, and ease of integration into current workflows. The two models are directly compared in this discussion, which also sheds light on their real-world uses and helps choose the best model depending on the particular needs and limitations of a given project.

5.4. Optimizer Impact on RNN Classification

The indistinguishable performance of Adam and SGD in this context underscores the robustness of the RNN architecture to optimizer variations, indicating that either can be effectively employed depending on specific computational or temporal constraints. The main conclusions of the study comparing Adam and SGD optimizers in an RNN configuration are succinctly summarized here, along with some useful applications for the two within related tasks.

6. Overall Evaluation of Attempts and Outcomes:

6.1. Evaluating the Effectiveness of Built Models in Achieving Their Intended Purpose:

The performance of these models demonstrates a strong ability to handle well-represented classes within the dataset, suggesting that they are particularly effective in scenarios where certain tags or categories dominate the data. However, the models' limited success in accurately identifying less frequent classes raises concerns about their generalizability and effectiveness across a more diverse set of NER tags. The models built, specifically SVM and RNN using TF-IDF, have successfully fulfilled their primary purpose of classifying NER tags with high accuracy (approximately 87.58%). They effectively identify the dominant class ('B-O') but struggle with less frequent classes, indicating good but not comprehensive performance.

6.2. Determining Acceptable F1/Accuracy Levels:

Considering the trade-offs and practical ramifications within the application area is necessary when setting a benchmark for 'good enough' F1-score/accuracy. The exact requirements and work context determine the F1-score and accuracy. An F1-score above 0.80 for dominating classes is generally regarded as effective for NER tasks. Both models achieved high F1-scores for the 'B-O' class but need improvement in handling minority classes.

6.3. Improving model performance in case of underperformance:

To enhance models struggling with minority classes, implementing techniques like SMOTE for rebalancing class distribution during training is crucial. Furthermore, refining feature extraction and optimizing model parameters can contribute to better performance. Additionally, exploring advanced neural network architectures or alternative embedding techniques such as Word2Vec or BERT may lead to improved representation and classification of less frequent classes.

6.4. Exploring trade-offs between model efficiency and quality for well-performing models:

Good efficiency is shown by the current models, especially the SVM with TF-IDF. In order to optimize efficiency even more, tactics like narrowing the feature space or streamlining the model architecture could be taken into consideration. To prevent overly sacrificing quality, it's imperative to strike a balance. Evaluating marginal feature reductions or transitioning to simpler models might assist in weighing the efficiency advantages against any performance drawbacks.

6.5. Justification of Choice Between Most Accurate vs. Most Effective Solution:

The SVM is a realistic option for deployment and interpretation, especially in circumstances with limited resources, despite the fact that both models performed equally. Even said, a more sophisticated model, like as an RNN, may perform more well for tasks requiring sophisticated language understanding, despite its higher complexity and computing requirements.

7. Conclusion:

Model Performance: The accuracy of the SVM and RNN models was similar, demonstrating their suitability for NER tasks.

Trade-offs: While the RNN delivers greater linguistic understanding at the expense of increasing computational complexity, the SVM model is simpler and easier to deploy and comprehend.

Handling Minority Classes: It appears that in order to enhance performance on these classes, methods such as data augmentation or more complex architectures are required, as both models had trouble with less frequent classes.

Resource Efficiency: The RNN may be appropriate for tasks demanding a deeper level of contextual understanding, whereas the SVM model, because to its computational lighter nature, may be favoured in contexts with limited resources.

Future Directions: To improve overall performance, more research may involve adjusting model parameters, experimenting with various feature extraction strategies, or using ensemble methods.