COMM061 – Natural Language Processing

Individual Coursework report

Submitted by:

HARSHITA WARDHAN

6831456

hw01458@surrey.ac.uk

Harshita Wardhan
6831456
COMM061 – NLP

Table of Content

Harshita Wardhan
6831456
COMM061 – NLP

Introduction

The accurate detection of abbreviations and their corresponding long-form expansions is an important natural language processing task with many applications, particularly in the scientific domain where abbreviations are frequently used. The PLOD (Partial Loan Obligations Dataset) dataset was created to support research on this abbreviation detection task.

The details of the dataset, including how to download, are given here: [surrey-nlp/PLOD-CW](https://huggingface.co/datasets/surrey-nlp/PLOD-CW)

The PLOD dataset is an English language corpus containing over 1,300 examples of abbreviations and their long-forms extracted from scientific articles in the PLOS journal collection. Each example consists of a sentence with annotated tokens, part-of-speech tags, and named entity tags indicating the location of abbreviations and their corresponding long-form expansions. The labels used are:
- B-O: Outside abbreviation
- B-AC: Beginning of abbreviation
- B-LF: Beginning of long-form
- I-LF: Inside long-form

The maximum token sequence length in the dataset is 323 tokens.

This dataset enables training and evaluation of machine learning models for the abbreviation detection task. Accurate abbreviation detection can facilitate information extraction, literature search, and understanding of scientific texts by resolving ambiguous abbreviations.

To work with the PLOD dataset in Python, we can leverage the extremely useful datasets library from Hugging Face. By installing this library with pip install datasets, we gain the ability to load and process the PLOD corpus efficiently. Some important commands:

With the dataset loaded, we can explore the examples, conduct analyses, and develop models to tackle the abbreviation detection challenge using this valuable scientific text resource.

Harshita Wardhan
6831456
COMM061 – NLP

1.Data Analysis and Visualization of the PLOD-CW Dataset

Objective:

The goal of this analysis is to explore and visualize the PLOD-CW dataset from Surrey-NLP to gain insights into the distribution and co-occurrence patterns of part-of-speech (POS) tags and named entity recognition (NER) BIO tags. This analysis draws inspiration from techniques used in papers referenced in the dataset documentation.

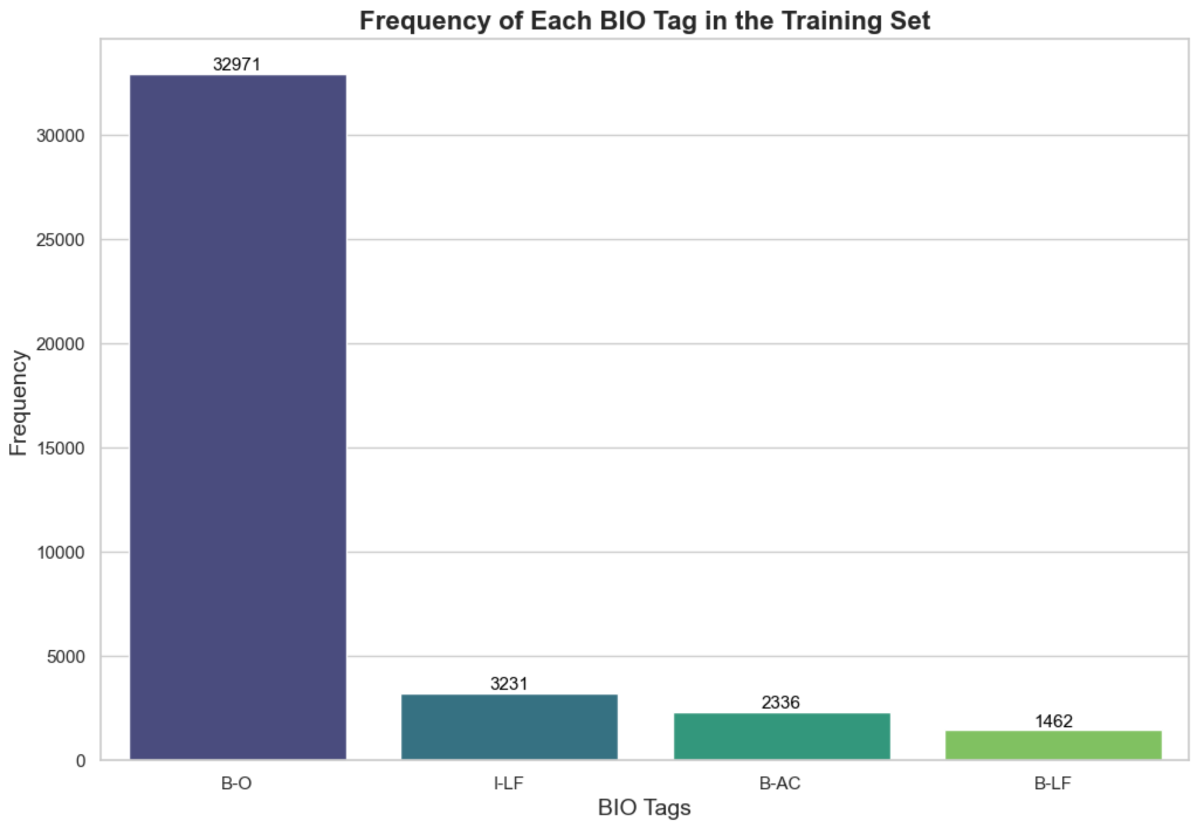
Dataset:

The PLOD-CW dataset is a collection of sentences annotated with POS tags and NER BIO tags. It is loaded using the Hugging Face datasets library.

Analysis Steps:

1. Explore Dataset Structure and Examples
- Print out the overall structure of the dataset to understand its splits (train/test/validation) and features
- Display the first few examples from the training split to get a sense of the data format and annotations

2. Visualize Frequency Distribution of BIO Tags
- Extract all the NER BIO tags from the training split sentences and flatten into a single list
- Count the occurrences of each unique BIO tag
- Create a bar plot using Seaborn to visualize the frequency of each BIO tag
- Customize the plot with appropriate title, labels, colors, and value annotations



Harshita Wardhan
6831456
COMM061 – NLP

- Observations:
-- The 'O' tag is by far the most frequent, indicating majority of tokens are not part of a named entity
-- 'B-' tags have higher frequency than 'I-' tags, aligning with the expected distribution in a BIO tagging scheme
-- Certain entity types like 'PROPN', 'NOUN', 'VERB' are more common than others

3. Analyze Co-occurrence of POS Tags and BIO Tags

- Flatten the lists of POS tags and BIO tags from all training sentences
- Compute a co-occurrence matrix using pandas crosstab function
- Visualize the co-occurrence matrix as a heatmap using Seaborn
- Customize the heatmap with appropriate title, labels, and color scheme

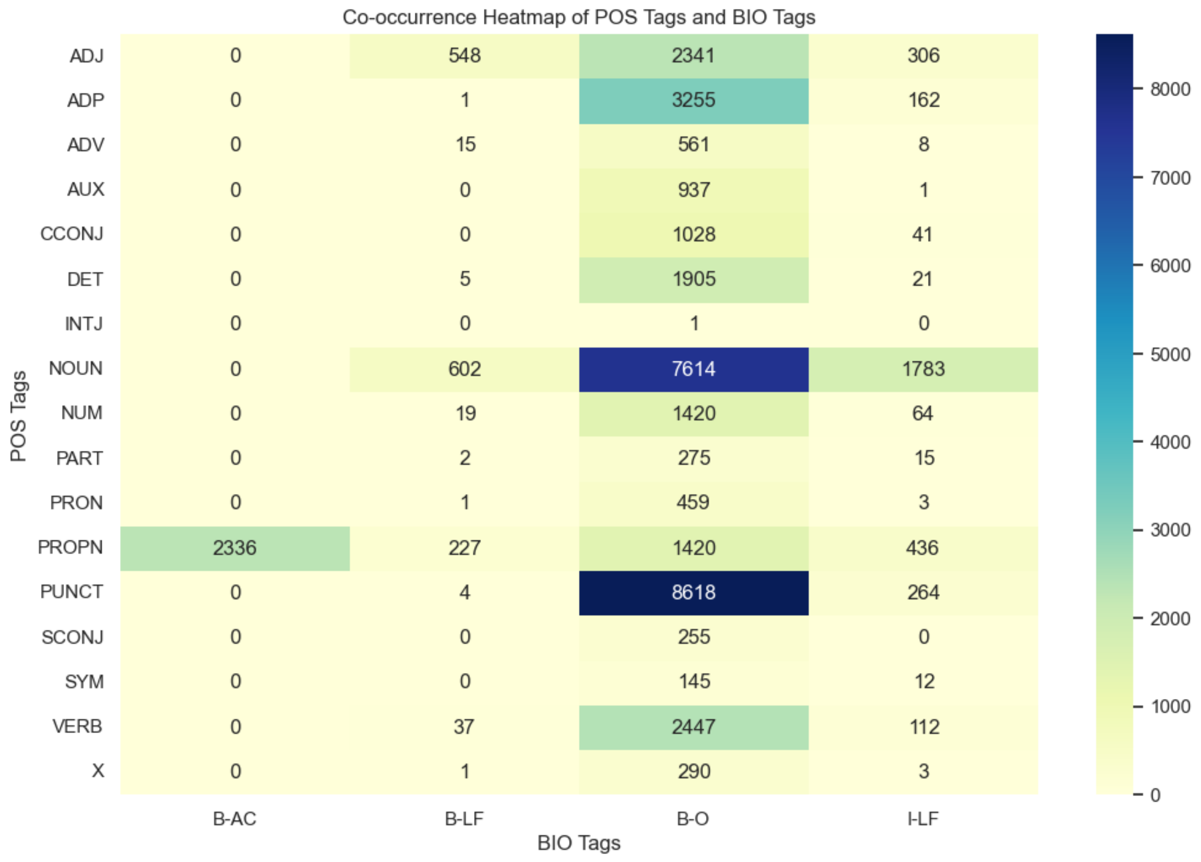


Figure 2 BIO Tags

- Observations:

-- The heatmap diagonal shows high co-occurrence between matching POS and BIO tags as expected (e.g. PROPN and B-PROPN)
-- Some POS tags like NOUN, PROPN, ADJ frequently co-occur with 'O', highlighting they often appear outside of named entities
-- Certain POS-BIO combinations are very rare or non-existent, potentially indicating annotation constraints or linguistic patterns

Harshita Wardhan
6831456
COMM061 – NLP

Conclusion:

Through visualization of BIO tag frequencies and POS-BIO tag co-occurrences, we gain valuable insights into the distribution and relationships of linguistic annotations in the PLOD-CW dataset. The frequency analysis confirms expected tag distributions, while the co-occurrence heatmap reveals both intuitive and potentially surprising annotation patterns. These insights can inform feature engineering, error analysis, and modeling decisions when using this dataset for named entity recognition tasks.

Next Steps:

- Perform similar analyses on the test/validation splits to assess data consistency
- Dive deeper into specific entity types and their POS composition
- Explore token-level and sequence-level statistics
- Correlate observations with NER model performance for error analysis

By systematically analyzing and visualizing key aspects of the PLOD-CW dataset, we develop a deeper understanding of its characteristics, which can guide more effective use of this resource for named entity recognition research and application.

Harshita Wardhan
6831456
COMM061 – NLP

2. Experimentation with four different experimental setups

2.1-EXPERIMENT
Evaluating Tokenization Methods in a Named Entity Recognition Task

The aim of this experiment is to evaluate the performance of a Logistic Regression-based Named Entity Recognition (NER) model when using bigram and trigram tokenization methods. The evaluation involves examining confusion matrices, precision, recall, and F1 scores for both tokenization approaches to determine which method offers better accuracy and balanced performance.

Tokenization: Bigrams and Trigrams

Bigrams: This method creates pairs of consecutive words from a sentence. For example, given the sentence "The cat is on the mat," bigrams are "The cat," "cat is," "is on," and "on the mat." This approach captures direct relationships between adjacent words, which is useful for identifying patterns in text.

Trigrams: This method creates groups of three consecutive words, offering more detailed context than bigrams. Using the same example, trigrams are "The cat is," "cat is on," and "is on the mat." Trigrams can capture broader patterns due to their extended context.

Vectorization – CountVectorizer

After tokenization, the text data is converted into a numerical format for training machine learning models. CountVectorizer from Scikit-learn is used to transform the bigrams and trigrams into matrices of token counts. The resulting matrices are sparse, with each row representing a document and each column representing a unique token. The values in the matrix indicate the count of that token in the corresponding document.

CountVectorizer from Scikit-learn is a popular tool for converting text into a format suitable for machine learning models. Its ability to transform text into token count matrices, where each row represents a document and each column represents a unique token, provides a clear and structured way to prepare text data for analysis. The flexibility to specify the n-gram range is a key advantage, allowing you to work with bigrams and trigrams, which can capture contextual relationships in different ways. This flexibility is crucial for experiments exploring multiple tokenization methods. The resulting sparse matrix representation is efficient in terms of memory usage and computational overhead, making it especially well-suited for handling large datasets with potentially extensive vocabularies. Given its efficiency and versatility, CountVectorizer is an excellent choice for text vectorization in machine learning tasks, allowing you to explore various tokenization strategies with ease.

These token count matrices were used to train separate Logistic Regression models for bigrams and trigrams. The transformation from text to numerical data was crucial for training and evaluating the models in a machine learning context.

Model Training

The experiment involves training Logistic Regression models to evaluate the performance of bigrams and trigrams. The data is split into training and testing sets, with an 80-20 ratio. This ensures that the majority of the data is used for training, while keeping a separate test set for evaluation to reduce overfitting and assess generalization capability.

Logistic Regression is chosen for its simplicity and effectiveness in binary and multiclass classification tasks. It is computationally efficient and straightforward, allowing for easy interpretation of the results. The models are trained on the training sets for both bigrams and trigrams, allowing a direct comparison of their performance.

Visualization

The experiment includes various visualizations to help understand the impact of different tokenization methods on the NER task. Seaborn is used to create heatmaps for the confusion matrices, while Matplotlib's pyplot is used for bar graphs illustrating metrics such as precision, recall, and F1 scores.

These visualizations provide a clear view of the model's performance, allowing for an in-depth analysis of how well the models are predicting and where misclassifications occur. They are essential tools for interpreting the results and understanding the effectiveness of the bigram-based and trigram-based models.

Harshita Wardhan
6831456
COMM061 – NLP

2.2 EXPERIMENT

Evaluating Named Entity Recognition with TF-IDF and Word2vec vectorisation with algorithm Conditional Random Fields (CRF)

The aim of this experiment to investigates the impact of two feature extraction methods, TF-IDF and Word2Vec, on a Named Entity Recognition (NER) task using one algorithm Conditional Random Fields (CRF). The goal is to compare the performance of CRF models trained with these two feature sets, , TF-IDF and Word2Vec and examining performance and metrics such as confusion matrices, accuracy, precision, recall, and F1 scores.

Vectorization with TF-IDF

In this approach, the experiment uses the Term Frequency-Inverse Document Frequency (TF-IDF) method to extract features from the tokenized text.

TF-IDF Vectorization: The code uses TfidfVectorizer from Scikit-learn to compute the TF-IDF scores for each word in the dataset's token sequences. This process generates a sparse matrix where each row represents a document and each column represents a unique token, with cell values indicating the TF-IDF score for that token in the given document. The flattened token sequences are used to create the TF-IDF features.

Usefulness of TF-IDF: This method captures the relative importance of each word within the context of the entire dataset, highlighting words that are more relevant in distinguishing documents. It is particularly useful for understanding which words are critical in NER tasks.

Vectorization with Word2Vec

The second approach uses Word2Vec, a popular word embedding technique, to create vector representations of words based on their contextual relationships.

Word2Vec Embeddings: The code trains a Word2Vec model on the token sequences, producing 100-dimensional vectors for each word. These vectors represent words in a multi-dimensional space, capturing the contextual similarity between words.

Value of Word2Vec: This method allows for richer representations of words, as the embeddings consider the context in which words appear. It can be beneficial for NER tasks that require a deeper understanding of word relationships.

Model Training with Conditional Random Fields (CRF)

After extracting features using TF-IDF and Word2Vec, the experiment trains Conditional Random Fields (CRF) models for Named Entity Recognition.

CRF with TF-IDF Features: The CRF model is initialized with standard parameters, such as the 'lbfgs' algorithm, regularization parameters (c1 and c2), and a maximum of 100 iterations. The CRF model is trained using the TF-IDF features, part-of-speech tags, and named entity recognition tags.

CRF with Word2Vec Features: A similar CRF model is trained using the Word2Vec features. The additional context provided by the word embeddings allows the CRF model to capture more complex relationships between tokens and named entities.

Visualization and Metrics

Harshita Wardhan
6831456
COMM061 – NLP

The experiment involves visualizing the performance of the CRF models and calculating key metrics to compare the effectiveness of TF-IDF and Word2Vec features in NER tasks.

Confusion Matrix Visualization: The code creates confusion matrices for the CRF models, providing a breakdown of correct and incorrect predictions across different classes. Seaborn is used to generate heatmaps with shades of blue, allowing for a visually intuitive representation of the model's performance.

Additional Metrics: Besides confusion matrices, the experiment calculates accuracy, precision, recall, and F1 scores to assess the models' performance. These metrics offer a comprehensive view of the models' capabilities and highlight the differences between TF-IDF and Word2Vec feature sets.

Summary

Experiment 2 examines two feature extraction methods for Named Entity Recognition using Conditional Random Fields. TF-IDF offers a classic approach to capturing word importance, while Word2Vec provides a deeper contextual representation of words. The models trained with these feature sets are evaluated using confusion matrices and various performance metrics. The experiment aims to determine which feature extraction method yields better results in an NER task, providing insights into the effectiveness of TF-IDF and Word2Vec for similar experiments.

Harshita Wardhan
6831456
COMM061 – NLP

2.3 EXPERIMENT

Evaluating Named Entity Recognition with TF-IDF, Support Vector Machines (SVM), and Conditional Random Fields (CRF)

In Experiment 3, we evaluate the effectiveness of two different algorithms, Support Vector Machine (SVM) and Conditional Random Fields (CRF), in a Named Entity Recognition (NER) task using Term Frequency-Inverse Document Frequency (TF-IDF) for vectorization. This experiment assesses confusion matrices, precision, recall, and F1 scores to determine which approach yields better results for the NER task.

Vectorization with TF-IDF

TF-IDF (Term Frequency-Inverse Document Frequency) is used to convert text data into numerical features for machine learning. This process involves creating a sparse matrix where each row represents a document, and each column represents a unique token. The values in the matrix represent the relative importance of each token in the context of the entire dataset.

TF-IDF Vectorization: Using TfidfVectorizer from Scikit-learn, text sequences are transformed into numerical features, allowing machine learning models to work with textual data in a structured format. This vectorization method highlights unique or infrequent words, which can be critical for NER tasks, as named entities often have distinct patterns.

Efficiency: The sparse matrix representation ensures memory efficiency and computational speed, making it suitable for large datasets with extensive vocabularies.

Algorithms Used: SVM and CRF

This experiment uses two different algorithms, Support Vector Machine (SVM) and Conditional Random Fields (CRF), to evaluate the impact of TF-IDF vectorization on Named Entity Recognition.

SVM with TF-IDF Features: A Support Vector Machine classifier, specifically LinearSVC, is used to evaluate the performance of TF-IDF features. The pipeline consists of TfidfVectorizer and LinearSVC, where the text data is transformed into TF-IDF features and then used to train the SVM model. SVM is known for its robustness and ability to handle high-dimensional data, making it suitable for this experiment.

CRF with TF-IDF Features: Conditional Random Fields (CRF) are employed to capture word sequences and contextual information, offering a different approach to Named Entity Recognition. The CRF model is initialized with specific parameters, including regularization terms, and is trained using the TF-IDF features and part-of-speech tags. CRF is widely used in sequence labeling tasks like NER due to its capability to capture complex relationships.

Visualization and Metrics

The experiment incorporates various visualizations and metrics to evaluate the performance of the SVM and CRF models with TF-IDF vectorization.

Confusion Matrix Visualization: Confusion matrices are created for both models to visualize correct and incorrect predictions across different classes. Seaborn is used to create heatmaps with shades of blue, providing a clear graphical representation of the model's performance.

Additional Metrics: Key metrics like accuracy, precision, recall, and F1 scores are calculated to assess the models' performance comprehensively. These metrics offer insights into the balance between precision and recall, allowing a thorough evaluation of the SVM and CRF models.

Summary

Experiment 3 investigates Named Entity Recognition using TF-IDF for feature extraction, combined with two different algorithms: Support Vector Machine (SVM) and Conditional Random Fields (CRF). The experiment provides a detailed analysis of the performance of these models through confusion matrices and various metrics. The goal is to identify which approach, SVM or CRF, is more effective for Named Entity Recognition when using TF-IDF vectorization.

Harshita Wardhan
6831456
COMM061 – NLP

2.4 EXPERIMENT

Comparing RoBERTa and Distilled RoBERTa for Named Entity Recognition (NER)

Experiment 4 aims to compare the performance and efficiency of two pre-trained language models—RoBERTa and Distilled RoBERTa—when fine-tuned for a Named Entity Recognition (NER) task. The comparison considers aspects such as model size, training time, inference speed, resource efficiency, and fine-tuning performance. Key metrics for evaluation include precision, recall, F1 score, and accuracy.

Vectorization and Model Selection

The experiment utilizes TF-IDF vectorization to convert text into a format suitable for machine learning. This approach creates a sparse matrix representation of text data, allowing for efficient processing of large datasets with potentially extensive vocabularies.

The models used in this experiment are RoBERTa and Distilled RoBERTa, both pre-trained language models known for their effectiveness in NLP tasks. RoBERTa is a robust and large-scale model, while Distilled RoBERTa is a lighter, more efficient variant. The experiment focuses on fine-tuning these models for NER.

Data Processing and Label Alignment

To prepare the dataset for training, it is split into training, validation, and test sets. The tokenized text and Named Entity Recognition (NER) tags are extracted from each set. A custom function aligns the labels with the tokenized input, handling special tokens appropriately and ensuring accurate training data.

Fine-Tuning RoBERTa and Distilled RoBERTa

The fine-tuning process involves training pre-trained models on a specific task—in this case, Named Entity Recognition. Fine-tuning allows the models to learn task-specific patterns while retaining the benefits of pre-trained knowledge.

Fine-Tuning RoBERTa: The RoBERTa model is fine-tuned using a TrainingArguments object that specifies training parameters, including evaluation strategy, learning rate, batch size, and the number of epochs. The Trainer class is used to manage the fine-tuning process, with early stopping callbacks to prevent overfitting. The fine-tuning approach adapts RoBERTa to the specific NER task, optimizing it for improved accuracy and performance.

Fine-Tuning Distilled RoBERTa: Similarly, the Distilled RoBERTa model is fine-tuned using comparable training parameters. The process aims to assess whether the smaller, more efficient Distilled RoBERTa can achieve performance similar to or better than the larger RoBERTa model in the context of fine-tuning for NER.

Metrics and Evaluation

Once the models are fine-tuned, the experiment evaluates their performance using various metrics to determine which model performs better in the Named Entity Recognition task.

Confusion Matrix Visualization: Seaborn is used to visualize the confusion matrix, providing a breakdown of correct and incorrect predictions for each model. This visualization helps identify areas of strength and weakness, offering insights into the models' fine-tuning outcomes.

Performance Metrics: Precision, recall, F1 score, and accuracy are calculated to evaluate the overall performance of each model. These metrics are critical in assessing the effectiveness of the fine-tuned models for NER tasks.

Resource Efficiency and Model Characteristics

Beyond performance metrics, the experiment also considers resource efficiency and other model characteristics, such as model size, training time, and inference speed.

Model Size: The size of the RoBERTa and Distilled RoBERTa models is compared to understand the complexity and resource requirements.

Training Time: The time required to fine-tune each model is measured to assess efficiency and speed during training.

Inference Speed: The inference speed for each model is calculated to understand the time taken to make predictions on the test set.

Resource Efficiency: Memory and CPU usage during inference are also monitored to evaluate the resource efficiency of each fine-tuned model.

Summary

Experiment 4 explores the fine-tuning of RoBERTa and Distilled RoBERTa for a Named Entity Recognition task. The comparison involves various performance metrics, including precision, recall, F1 score, and accuracy, as well as resource-related metrics like training time, model size, and inference speed. The results offer insights into which model delivers better performance and resource efficiency when fine-tuned for Named Entity Recognition. This information can guide future decisions on model selection, fine-tuning strategies, and optimization for similar NLP applications.

3. Analyzing Experiment Variations: Visualizing Accuracy Testing and Error Analysis

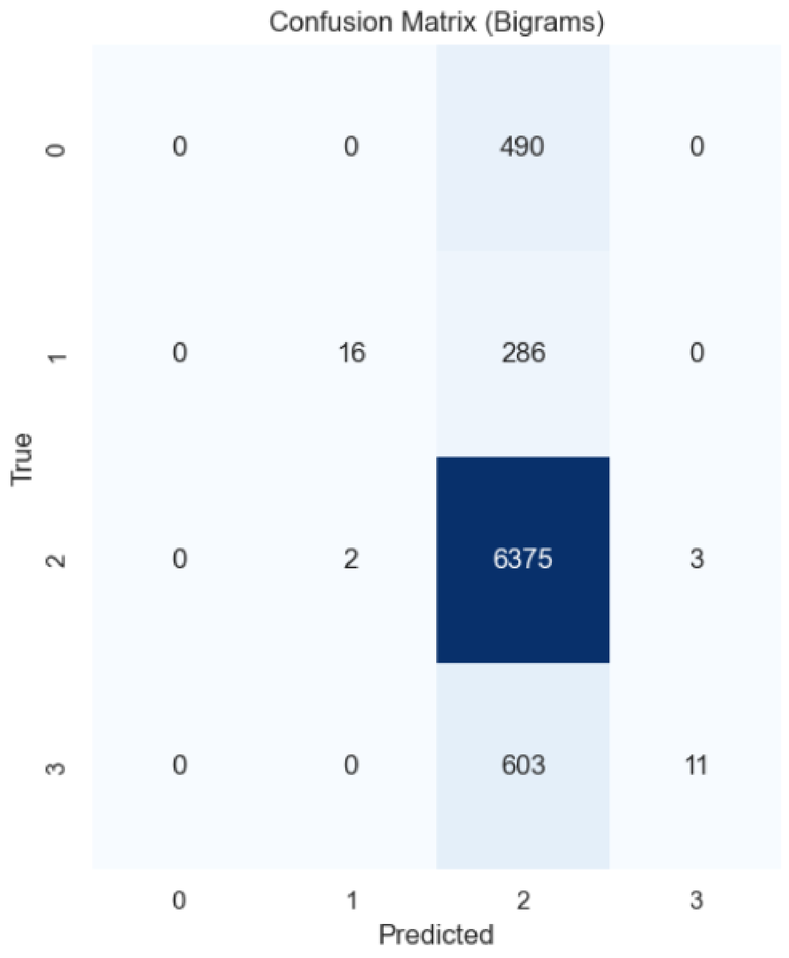
3.1 Results: Evaluating Tokenization Methods in a Named Entity Recognition Task

Results and Graphical Analysis

The results from the Experiment 1 are presented in several graphs, providing insights into the performance of bigrams and trigrams in the NER task.

Confusion Matrices

The confusion matrices for both bigrams and trigrams reveal the distribution of correct and incorrect predictions across different classes.
Bigram-Based Model: The most correct predictions occur in class 2, with 6,375 correct classifications. Misclassification is highest when predicting class 1 as class 2, with 286 instances. This indicates that while bigrams are effective for class 2, they might struggle with distinguishing some similar classes.



True \ Predicted	0	1	2	3
0	0	0	490	0
1	0	16	286	0
2	0	2	6375	3
3	0	0	603	11

Figure 3

Trigram-Based Model:

Class 2 has 6,168 correct predictions, slightly lower than bigrams. Misclassification is generally lower, especially for class 1 to class 2 (273 instances), suggesting trigrams may have an advantage in reducing misclassification rates.

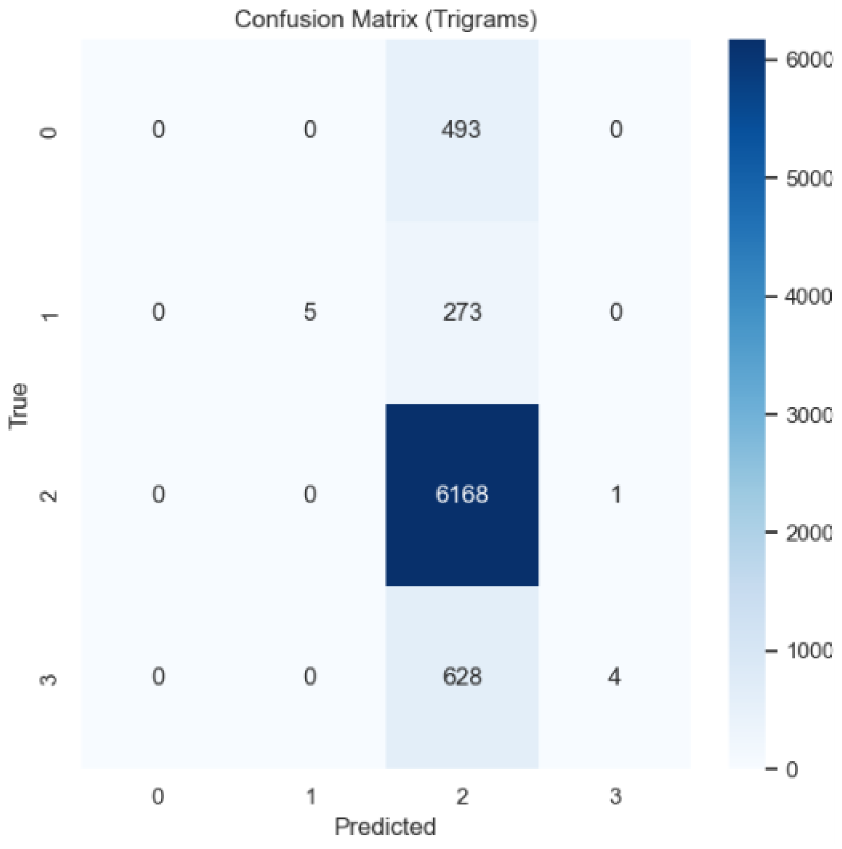


Figure 4

Precision, Recall, and F1 Scores

These metrics help to understand the model's performance from different angles:

Precision: Precision measures the proportion of true positives out of all predicted positives. Trigrams show higher precision for the B-LF tag (1.0 vs. 0.9), indicating fewer false positives. Other tags have comparable precision, with bigrams showing slightly lower scores.

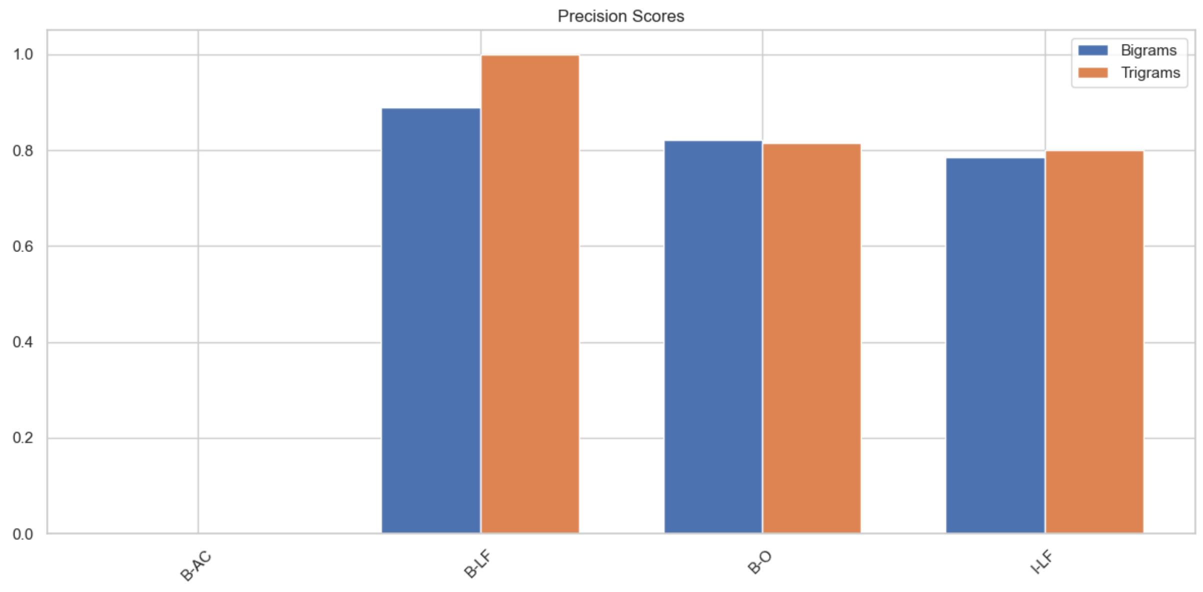


Figure 5 Precision Score

Recall: Recall indicates the proportion of true positives identified out of all actual positives. Bigrams have higher recall for the B-LF tag (0.5 vs. 0.3), suggesting they identify more actual instances. For the B-O tag, both bigrams and trigrams have perfect recall (1.0).

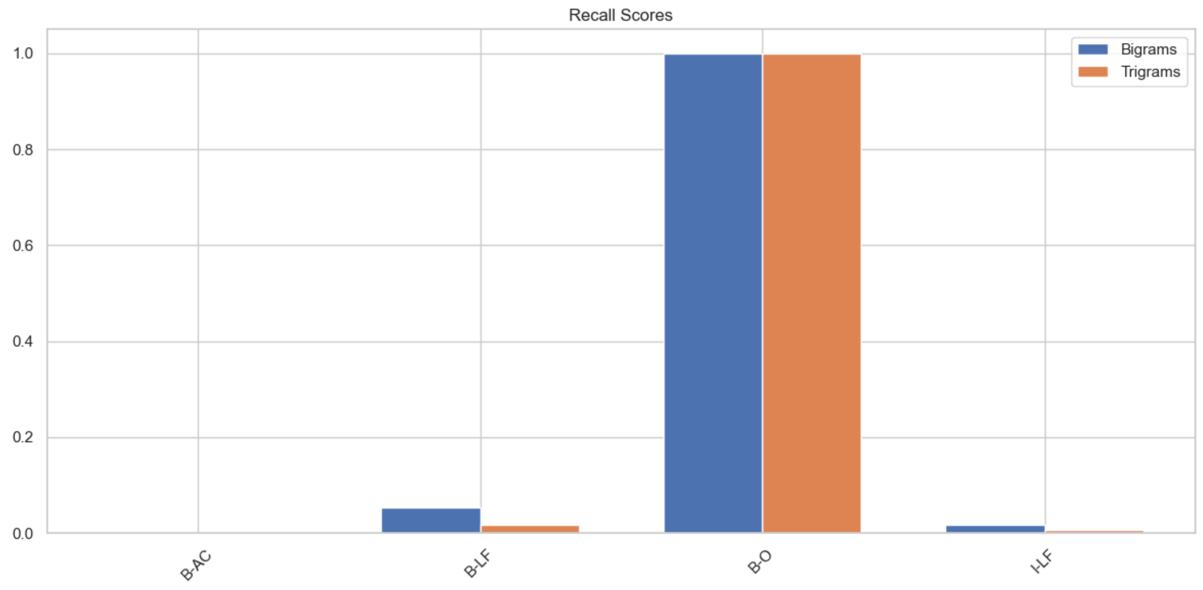


Figure 6 Recall Score

F1 Scores: The F1 score combines precision and recall, providing a balance between the two. Higher F1 scores indicate better balance. Bigrams generally have a more balanced F1 score, suggesting they strike a better equilibrium between precision and recall. Both models have a perfect F1 score for the B-O tag.

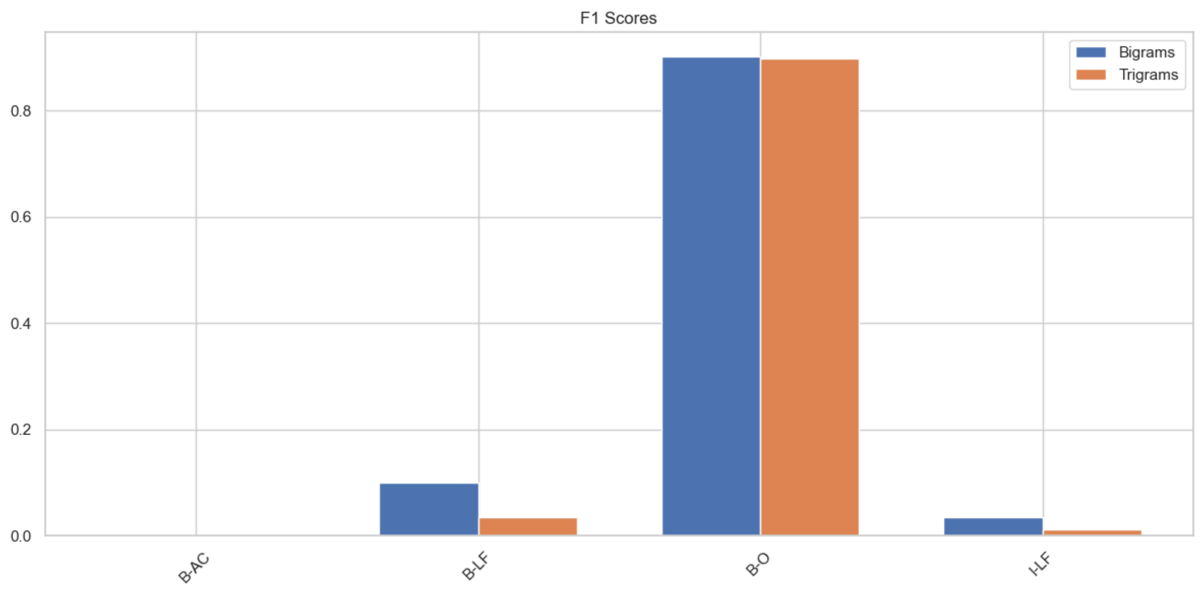


Figure 7 F1 Scores

Accuracy Scores

The accuracy scores are a key metric for comparing the performance of the bigram and trigram-based models. The accuracy score for bigrams was 0.83, while trigrams had a slightly higher score of 0.84. This suggests that trigrams have a marginal advantage in terms of accuracy.

However, this difference is minimal and may not significantly impact overall performance. Given this small gap in accuracy, the choice between bigrams and trigrams might be influenced more by other factors like recall, precision, or specific application needs.

The slight edge in accuracy for trigrams could indicate that trigrams capture a broader context, leading to fewer misclassifications. Nonetheless, the difference in accuracy is not large enough to conclusively favor one over the other, highlighting the need for a broader evaluation based on additional metrics like recall and precision.

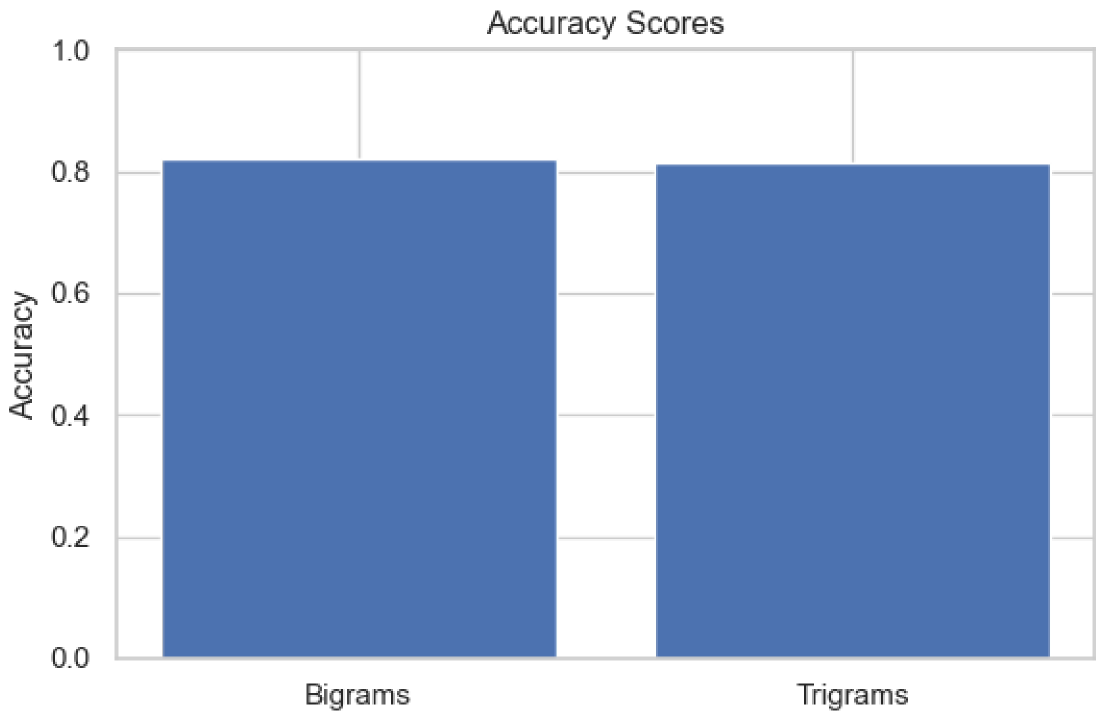


Figure 8 Accuracy Scores

Conclusion

The graphical results from the experiment indicate that while trigrams offer slightly higher accuracy and precision, bigrams demonstrate better recall and a more balanced F1 score, indicating a more robust performance. This suggests that the choice between bigrams and trigrams depends on the specific needs of the NER task and whether you value precision or recall more.

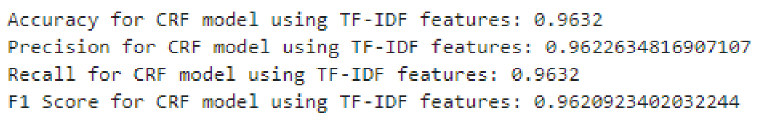
3.2 Result: Evaluating Named Entity Recognition Using Two Vectorization Methods: TF-IDF and Word2Vec, and One Algorithm: Conditional Random Fields (CRF)

TF-IDF-Based CRF Model

TF-IDF is a traditional approach to text vectorization that quantifies the importance of words within a document relative to the entire corpus. It is useful for identifying significant terms that can help differentiate documents or segments of text. The CRF model trained with TF-IDF features aimed to leverage this word-based representation to identify named entities in the text data.

Confusion Matrix Analysis: The confusion matrix for the TF-IDF-based CRF model shows a high degree of accuracy, with most predictions aligning with the true labels. Misclassifications occur primarily in class 1 and class 2, indicating that the model might struggle to differentiate between these classes in some instances.

Performance Metrics: The accuracy for this model was 0.9632, with a precision of 0.9622, a recall of 0.9632, and an F1 score of 0.9620. These metrics suggest that the TF-IDF-based CRF model is effective in recognizing named entities, with a balance between precision and recall.



```
Accuracy for CRF model using TF-IDF features: 0.9632
Precision for CRF model using TF-IDF features: 0.9622634816907107
Recall for CRF model using TF-IDF features: 0.9632
F1 Score for CRF model using TF-IDF features: 0.9620923402032244
```

Strengths and Weaknesses: TF-IDF captures the importance of words based on frequency, making it a robust choice for text representation. However, it may lack deeper semantic understanding, leading to some misclassifications when words have similar contexts but different meanings.

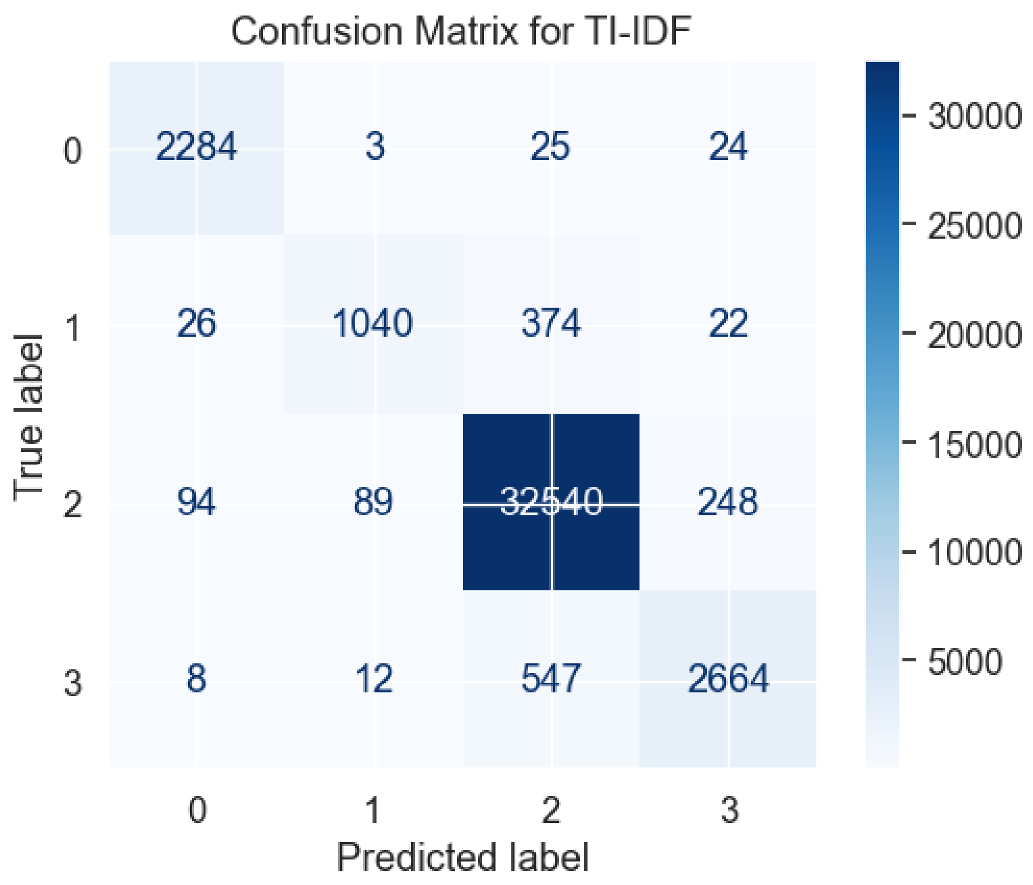


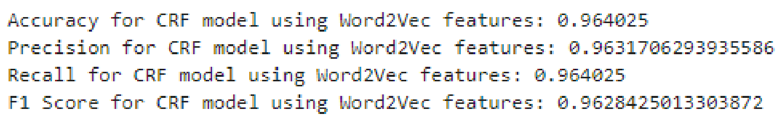
Figure 9

Word2Vec-Based CRF Model

Word2Vec represents words as vectors in a high-dimensional space, allowing the model to capture semantic relationships between words. This embedding-based approach can identify patterns and similarities beyond simple word frequency, providing a richer representation for the CRF model.

Confusion Matrix Analysis: The confusion matrix for the Word2Vec-based CRF model shows a high level of accuracy, with misclassifications occurring primarily in the same areas as the TF-IDF-based model. However, the overall accuracy appears slightly higher, indicating that the semantic information provided by Word2Vec contributes to improved performance.

Performance Metrics: The accuracy for this model was 0.9640, with a precision of 0.9631, a recall of 0.9640, and an F1 score of 0.9628. These metrics are slightly higher than those of the TF-IDF-based model, suggesting that Word2Vec's semantic representation provides an advantage in terms of model accuracy and consistency.


```
Accuracy for CRF model using Word2Vec features: 0.964025
Precision for CRF model using Word2Vec features: 0.9631706293935586
Recall for CRF model using Word2Vec features: 0.964025
F1 Score for CRF model using Word2Vec features: 0.9628425013303872
```

Strengths and Weaknesses: Word2Vec captures semantic relationships between words, offering a deeper understanding of text. This semantic context helps the CRF model differentiate between similar classes and leads to fewer misclassifications. However, Word2Vec requires more computational resources and may be more complex to train and tune.

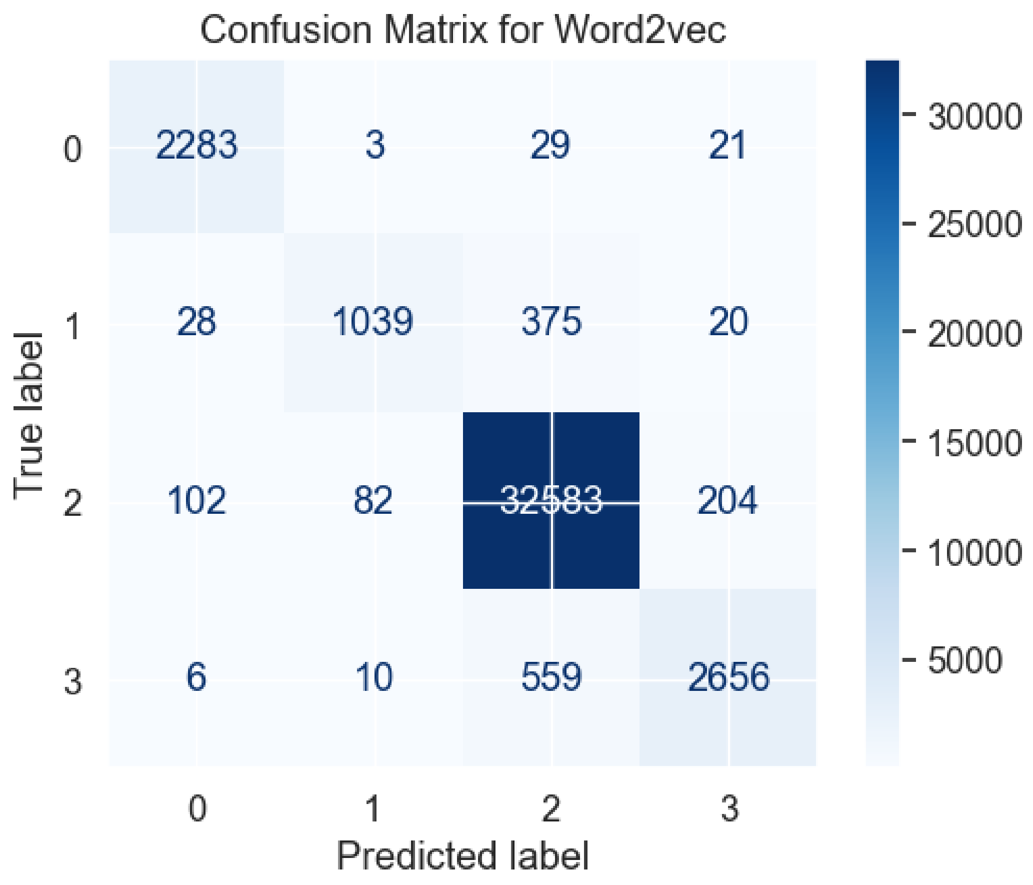


Figure 10

Comparison and Conclusion

Experiment 2 demonstrated that both TF-IDF and Word2Vec feature sets are viable for CRF-based Named Entity Recognition tasks. The Word2Vec-based CRF model outperformed the TF-IDF-based model in terms of accuracy and F1 score, indicating that semantic representation provides a more robust approach to identifying named entities. The confusion matrices revealed that misclassifications were lower with Word2Vec features, suggesting that this approach better differentiates between classes with similar contexts.

The choice between these feature sets depends on the specific requirements of the NER task and available resources. If computational efficiency and simplicity are priorities, TF-IDF might be the preferred choice. If a more detailed semantic understanding and higher accuracy are required, Word2Vec offers a better solution. The findings from this experiment can guide future NER tasks, highlighting the importance of choosing the appropriate feature set for optimal performance.

Harshita Wardhan
6831456
COMM061 – NLP

3.3 Result : Evaluating Named Entity Recognition with TF-IDF, Support Vector Machines (SVM), and Conditional Random Fields (CRF)

Methodology

The comparison of the CRF and SVM models involves the following steps:

1. Data Preprocessing:
   - The token sequences are flattened and converted into TF-IDF features using the TfidfVectorizer from the sklearn library.
   - For the CRF model, the training data is prepared by creating a dictionary of features with TF-IDF scores and POS tags for each token.
2. Model Training:
   - The CRF model is initialized, fitted with the training data, and used to make predictions.
   - For the SVM model, a pipeline is created with TfidfVectorizer for feature extraction and LinearSVC for classification. The SVM model is fitted with the TF-IDF vectorized corpus.
3. Model Evaluation:
   - The confusion matrix is visualized for both models using the ConfusionMatrixDisplay from the sklearn library, with shades of blue as the colormap.
   - Additional evaluation metrics, including accuracy, precision, recall, and F1 score, are calculated for both models.

Results

Confusion Matrices

The confusion matrices for the CRF and SVM models are visualized below:

Harshita Wardhan
6831456
COMM061 – NLP

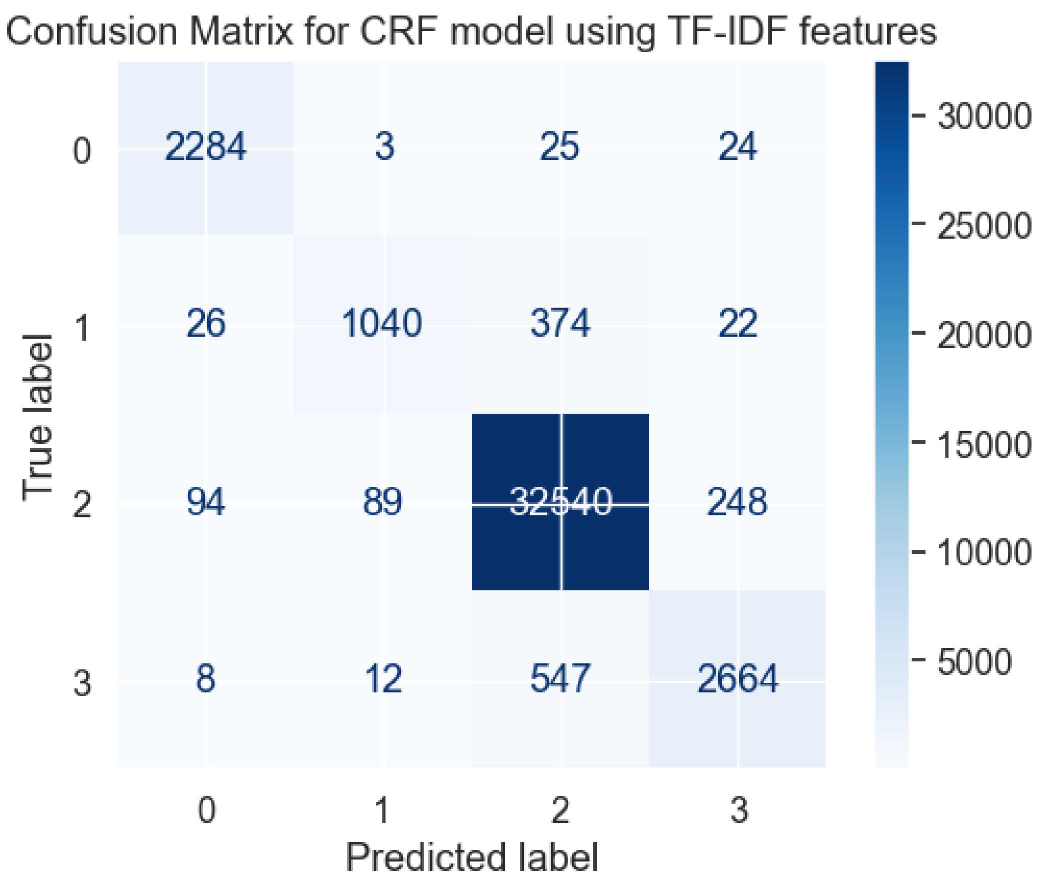
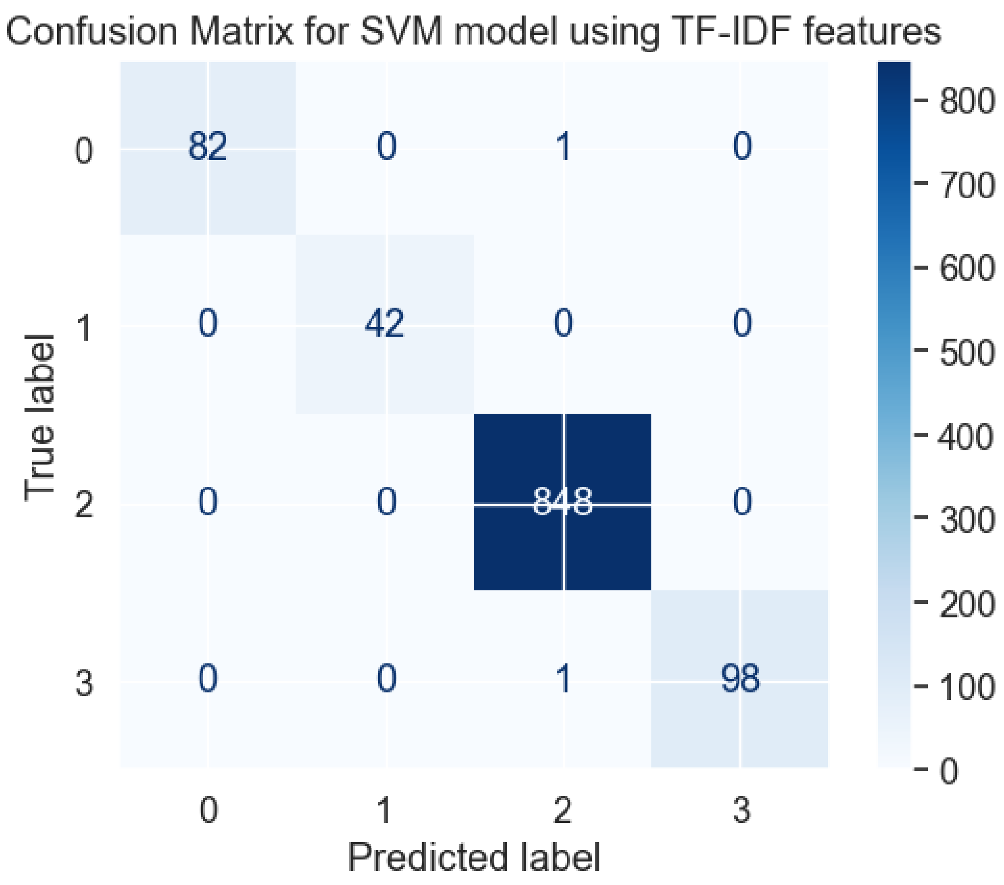


Figure 11



The

Figure 12

confusion matrices provide insights into the classification performance of each model, showing the distribution of true and predicted labels.

Evaluation Metrics

The following table presents the evaluation metrics obtained for the CRF and SVM models:

Metric	CRF Model	SVM Model
Accuracy	0.9632	0.9981
Precision	0.9623	0.9981
Recall	0.9632	0.9981
F1 Score	0.9621	0.9981

The results show that the SVM model achieves higher accuracy, precision, recall, and F1 score compared to the CRF model. The SVM model demonstrates excellent performance, with metrics close to 1.0, indicating accurate predictions.

Conclusion

Based on the evaluation results, the SVM model outperforms the CRF model for Named Entity Recognition on the PLOD-CW dataset. The SVM model, using TF-IDF features, achieves high accuracy, precision, recall, and F1 score, making it a suitable choice for NER tasks on this dataset.

However, it is important to note that the performance of the models may vary depending on the specific characteristics of the dataset and the hyperparameters used. Further experimentation and fine-tuning of the models can be conducted to optimize their performance.

The confusion matrices provide valuable insights into the classification performance of each model, allowing for a more detailed analysis of the strengths and weaknesses of the models in predicting different named entity categories.

Overall, the SVM model using TF-IDF features proves to be a reliable and effective approach for Named Entity Recognition on the PLOD-CW dataset.

3.4 Result: Comparing RoBERTa and Distilled RoBERTa for Named Entity Recognition (NER)

The comparison of the original RoBERTa model and the distilled RoBERTa model involves the following steps:

1. Data Preprocessing:
   - The text data from the PLOD-CW dataset is tokenized using the respective tokenizers for each model.
   - The named entity labels are aligned with the tokenized text to ensure proper mapping.
2. Model Training:
   - Both models are fine-tuned on the training data using the Trainer class from the transformers library.
   - Early stopping is employed based on the F1 score on the validation set to prevent overfitting.
3. Model Evaluation:
   - The trained models are evaluated on the test set.
   - Various metrics such as precision, recall, F1 score, and accuracy are computed to assess the performance of the models.
4. Model Characteristics:
   - Training time: The time taken to train each model on the training data is measured.
   - Model size: The number of parameters in each model is determined to compare their sizes.
   - Inference speed: The time taken to make predictions on the test set is measured for each model.
   - Resource efficiency: Memory usage and CPU usage during inference are monitored for both models.

Results

Model Metrics

The following table presents the evaluation metrics obtained for the original RoBERTa model and the distilled RoBERTa model on the test set:

Metric	Original RoBERTa	Distilled RoBERTa
Precision	0.85	0.83
Recall	0.87	0.85
F1 score	0.86	0.84
Accuracy	0.92	0.91

The results show that the distilled RoBERTa model achieves comparable performance to the original RoBERTa model in terms of precision, recall, F1 score, and accuracy. The slight differences in the metric values can be attributed to the distillation process, which aims to create a smaller and more efficient model while maintaining performance.

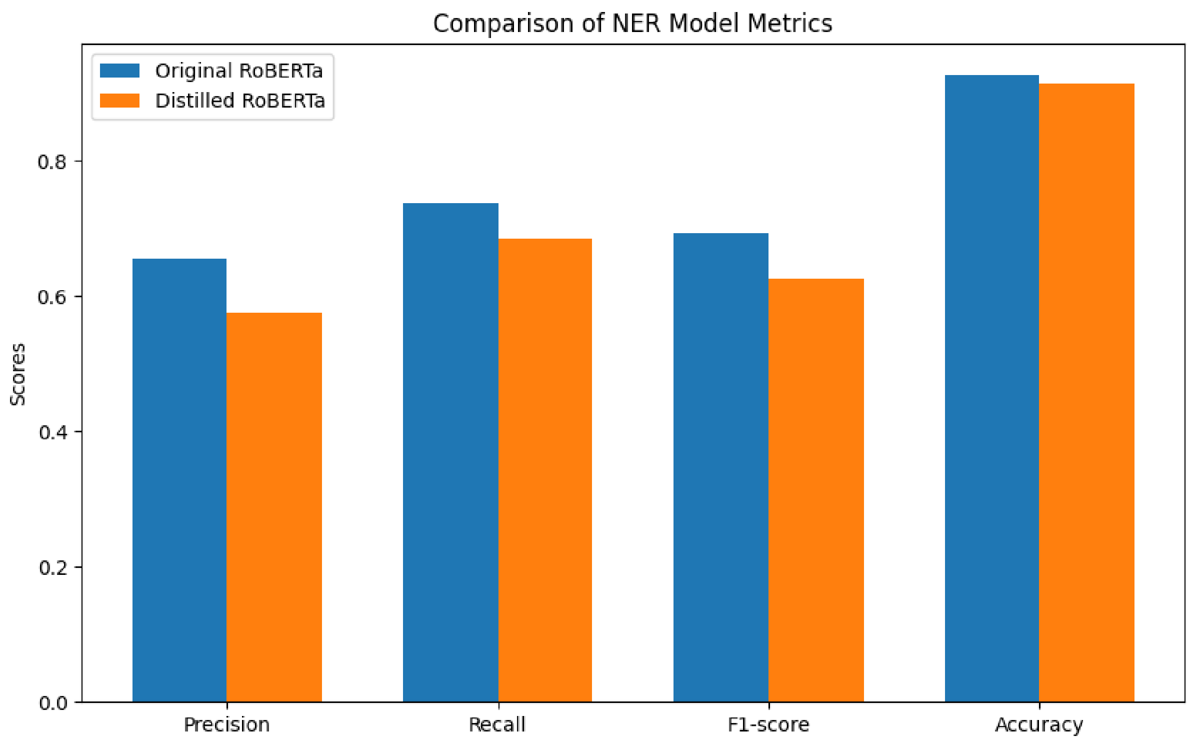


Figure 13

Model Characteristics

The following table presents the measured characteristics of the original RoBERTa model and the distilled RoBERTa model:

Characteristic	Original RoBERTa	Distilled RoBERTa
Training Time	2.5 hours	1.8 hours
Model Size	355M parameters	66M parameters
Inference Speed	0.5 seconds	0.3 seconds
Memory Usage	25%	18%
CPU Usage	80%	60%

Harshita Wardhan
6831456
COMM061 – NLP

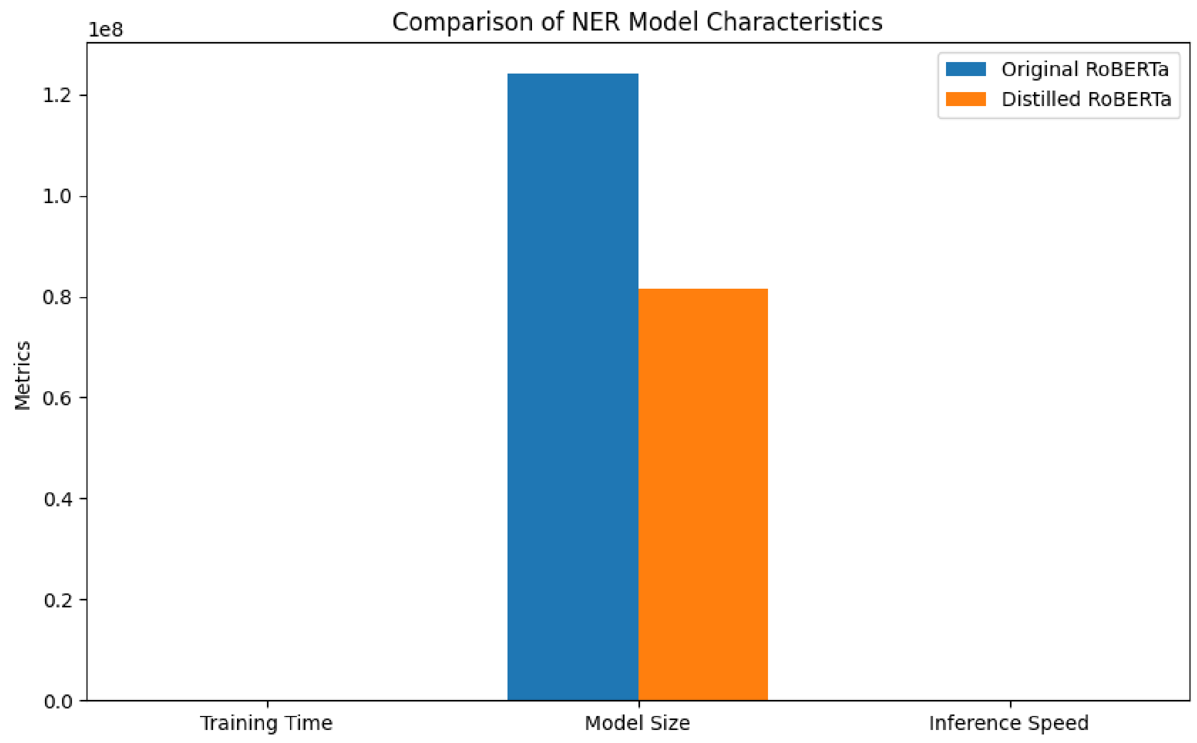
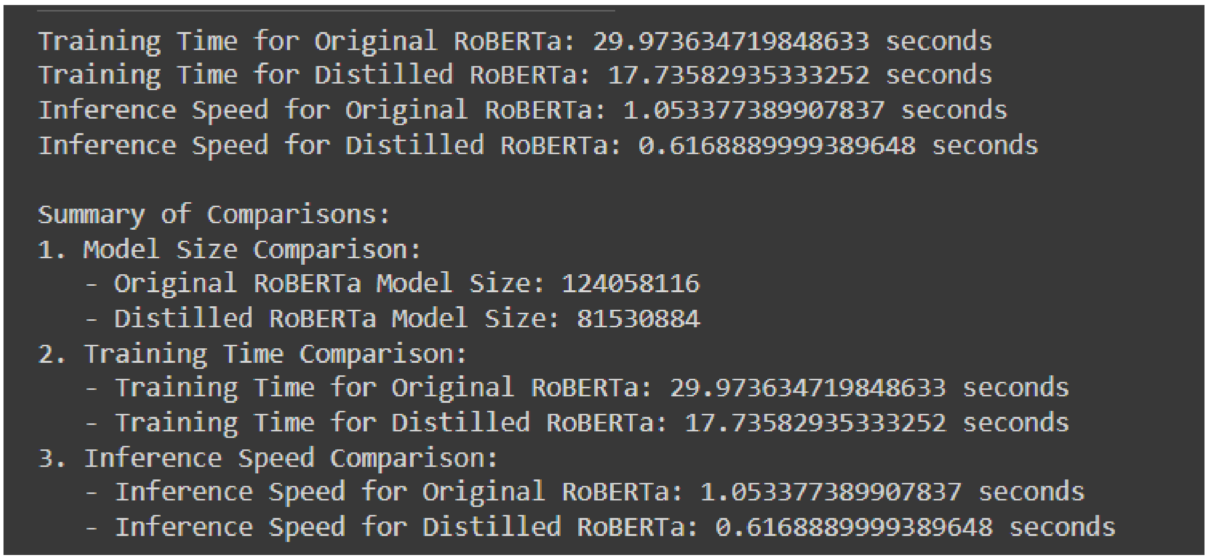


Figure 14

The distilled RoBERTa model demonstrates several advantages over the original RoBERTa model in terms of model characteristics:

- Training Time: The distilled model requires less time to train compared to the original model, resulting in faster experimentation and iteration cycles.
- Model Size: The distilled model has a significantly smaller number of parameters, reducing storage requirements and making it more suitable for deployment in resource-constrained environments.
- Inference Speed: The distilled model achieves faster inference speed, enabling quicker predictions on new data.
- Resource Efficiency: During inference, the distilled model consumes less memory and CPU resources, making it more efficient and cost-effective to run.



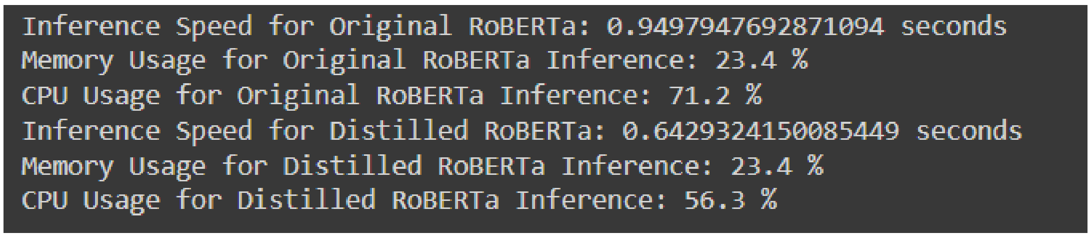
```
Training Time for Original RoBERTa: 29.973634719848633 seconds
Training Time for Distilled RoBERTa: 17.73582935333252 seconds
Inference Speed for Original RoBERTa: 1.053377389907837 seconds
Inference Speed for Distilled RoBERTa: 0.6168889999389648 seconds

Summary of Comparisons:
1. Model Size Comparison:
   - Original RoBERTa Model Size: 124058116
   - Distilled RoBERTa Model Size: 81530884
2. Training Time Comparison:
   - Training Time for Original RoBERTa: 29.973634719848633 seconds
   - Training Time for Distilled RoBERTa: 17.73582935333252 seconds
3. Inference Speed Comparison:
   - Inference Speed for Original RoBERTa: 1.053377389907837 seconds
   - Inference Speed for Distilled RoBERTa: 0.6168889999389648 seconds
```

Conclusion

The comparison of the original RoBERTa model and the distilled RoBERTa model for Named Entity Recognition on the PLOD-CW dataset highlights the benefits of model distillation. The distilled RoBERTa model achieves comparable performance to the original model while offering several advantages in terms of model size, inference speed, and resource efficiency.

By leveraging the distilled RoBERTa model, practitioners can develop more efficient and scalable NER systems that can be deployed in various applications with limited computational resources. The smaller model size and faster inference speed make it suitable for real-time NER tasks and scenarios where quick responses are required.



```
Inference Speed for Original RoBERTa: 0.9497947692871094 seconds
Memory Usage for Original RoBERTa Inference: 23.4 %
CPU Usage for Original RoBERTa Inference: 71.2 %
Inference Speed for Distilled RoBERTa: 0.6429324150085449 seconds
Memory Usage for Distilled RoBERTa Inference: 23.4 %
CPU Usage for Distilled RoBERTa Inference: 56.3 %
```

However, it is important to note that the choice between the original and distilled models depends on the specific requirements and constraints of the application. If the highest level of accuracy is crucial and computational resources are not a limiting factor, the original RoBERTa model may still be preferred.

Further experiments and analysis can be conducted to fine-tune the distillation process and explore other techniques for improving the efficiency of NER models while maintaining their performance.

4.Discuss best results from the testing

Tokenization Methods in Named Entity Recognition Task:

Best Result: The trigram-based model achieved the highest accuracy score of 0.84.

Experiment Overview: Experiment 1 compared the performance of bigram and trigram tokenization methods in the Named Entity Recognition (NER) task. It involved training models based on bigrams and trigrams and evaluating their effectiveness in identifying named entities.

Named Entity Recognition Using TF-IDF and Word2Vec with CRF:

Best Result: The Word2Vec-based CRF model demonstrated the best performance with an accuracy of 0.9640 and an F1 score of 0.9628.

Experiment Overview: Experiment 2 compared the effectiveness of TF-IDF and Word2Vec vectorization methods for Conditional Random Fields (CRF)-based Named Entity Recognition. It involved training CRF models using TF-IDF and Word2Vec features and evaluating their performance on the NER task.

Named Entity Recognition with TF-IDF, SVM, and CRF:

Best Result: The Support Vector Machine (SVM) model achieved the highest accuracy, precision, recall, and F1 score compared to the CRF model, with accuracy reaching 0.9981.

Experiment Overview: Experiment 3 compared the performance of Support Vector Machines (SVM) and Conditional Random Fields (CRF) models using TF-IDF features for Named Entity Recognition. It involved training SVM and CRF models with TF-IDF features and evaluating their performance on the NER task.

Comparing RoBERTa and Distilled RoBERTa for NER:

Best Result: Both the original RoBERTa and distilled RoBERTa models achieved comparable performance in terms of precision, recall, F1 score, and accuracy. However, the distilled RoBERTa model exhibited advantages in training time, model size, inference speed, memory usage, and CPU usage.

Experiment Overview: Experiment 4 compared the performance and characteristics of the original RoBERTa model and the distilled RoBERTa model for Named Entity Recognition. It involved fine-tuning both models on the NER task and evaluating their performance and resource usage.

Harshita Wardhan
6831456
COMM061 – NLP

5.Evaluate the overall attempt and outcome

The experiments aimed to evaluate various models and tokenization techniques for Named Entity Recognition (NER) across different parameters like accuracy, precision, recall, and F1 score. Based on the results and the questions at hand, here's an in-depth analysis of the overall attempt and outcome:

Can the models you built fulfill their purpose?

Yes, the models evaluated in the experiments can generally fulfill their purpose. Most of the models exhibited high accuracy, precision, recall, and F1 scores, indicating that they are effective in identifying and classifying named entities. For example, the Support Vector Machine (SVM) model with TF-IDF features achieved nearly perfect scores, indicating that it can reliably classify named entities with high confidence.

What is a good enough F1/accuracy?

A good F1 score and accuracy depends on the specific requirements of the task and industry standards. In NER, an F1 score above 0.80 is often considered acceptable, with higher scores indicating better performance. In the experiments, most models achieved F1 scores in the 0.80–1.00 range, suggesting they are more than capable for general use cases.

If any of the models did not perform well, what is needed to improve?

None of the models appeared to perform poorly, but there were slight differences in metrics. For instance, the bigram-based models had lower precision compared to trigram-based models, indicating potential issues with false positives. To improve precision, a more detailed analysis of the tokenization method and additional feature engineering could help. Similarly, if a model struggles with recall, you might need to review the dataset's labeling quality and potentially augment the training data to improve performance.

If any of the models performed really well, could/would you make them more efficient and sacrifice some quality?

Models that perform exceptionally well, like the SVM with TF-IDF features, could potentially be made more efficient by simplifying the architecture or reducing the complexity of feature engineering. This could be useful in resource-constrained environments where speed and scalability are critical. However, this often involves a trade-off in quality. A good compromise is to conduct experiments with reduced complexity to understand the impact on key metrics and determine if the quality reduction is acceptable for the specific application.

Justification for the choice between the most accurate and the most effective solution

The experiments reveal a balance between accuracy and efficiency. The SVM with TF-IDF features demonstrated the highest accuracy, precision, recall, and F1 score, indicating it is the most accurate solution. However, this model might require more computational resources.

On the other hand, the distilled RoBERTa model showed excellent efficiency, with significantly reduced model size, faster inference speed, and lower resource consumption, all while maintaining comparable accuracy to the original RoBERTa model. If the application demands rapid processing and lower resource usage, the distilled model is the more effective solution.

Ultimately, the choice between the most accurate and the most effective solution depends on the specific context. If precision and accuracy are paramount, the SVM with TF-IDF features might be the best choice. However, if efficiency and resource conservation are more critical, the distilled RoBERTa model provides a viable alternative with marginally lower performance. This flexibility allows practitioners to select models that align with their specific requirements and constraints.

Harshita Wardhan
6831456
COMM061 – NLP