Natural Language Processing (COMM061) -Individual Report

Name: Wilson Thomas Ayyappan
URN: 6835716
Group Number: 37

Table of Contents

1. Introduction

Sequence classification and labelling play crucial roles in the rapidly advancing field of Natural Language Processing (NLP), facilitating the extraction of valuable insights from text data. These techniques find applications across diverse domains such as security systems and healthcare monitoring, where timely and accurate data analysis is paramount. This coursework focuses on employing sequence classification to detect abbreviations and their expanded forms within texts, leveraging a labelled dataset sourced from scientific literature, specifically biomedical articles from the PLOS journal.

The complexity of scientific vocabulary, often spanning multiple words and exhibiting hierarchical structures, presents challenges for sequence classification, necessitating sophisticated labelling methods like the BIO format. This format employs tags like 'B-LF' to denote the start of a long form, 'I-LF' for its continuation, and 'B-O' for tokens unrelated to any abbreviation or long form. While the dataset used includes optional part-of-speech annotations, the primary focus remains on identifying lengthy forms and abbreviations.

This article delves into the various approaches undertaken to tackle the challenges posed by the PLOD dataset, employing a range of machine learning models and techniques. It explores different aspects of NLP problems, encompassing tokenization, utilization of diverse NLP methodologies, and model tuning to enhance prediction accuracy. Apart from successfully recognizing abbreviations and their expansions, the objective is to evaluate the effectiveness and efficiency of the models, striving to strike a balance between computational complexity and predictive performance.

The objective of this task is to document the process and results of developing a sequence classifier prototype for performing sequence classification on the PLOD dataset. The dataset contains 50k labelled tokens, and the aim is to use the BIO format labelling schema to classify them as abbreviations (labelled as AC) or long forms (labelled as LF).

2. Exploring and Visualizing the Dataset

2.1 Dataset Source and Features

- The Dataset is sourced from https://huggingface.co/datasets/surrey-nlp/PLOD-CW
- Labels: B-O, B-AC, B-LF, I-LF
- Maximum token sequence length in the dataset: 323

Features breakdown:

- **Tokens** - These are discrete words or symbols that were taken out of the dataset texts.
- **Part-of-speech tags (POS_tags)** - These are the grammatical information that are allocated to each token.
- **Named entity recognition tags (NER_tags)** - These tags group names, locations, organizations, and other attributes into predetermined categories for token classification.

2.2 Data Overview

The PLOD-CW dataset from the Surrey NLP Group is used to classify tokens, specifically entities, in English text. This dataset contains between 100,000 and 1 million tokens and is divided into training, validation, and test sets, totalling 1.35k rows. It consists of tokens, named entity recognition (NER) tags, and part-of-speech (POS) tags organized according to the BIO scheme. The primary goal of annotation is to identify abbreviations ('AC') and their equivalent extended forms ('LF'), which are critical for understanding scientific papers. Hadeel Saadany and Leonardo Zilio lead a team that thoroughly curated and annotated the dataset.

A preliminary analysis of the dataset, displayed as a Pandas Data Frame, indicates an organized format with three key columns: 'tokens', 'pos_tags', and 'ner_tags'. The 'tokens' column indicates individual words or symbols retrieved from the text and used to conduct our research. The 'pos_tags' provide syntactic classification for each token, revealing grammatical responsibilities inside phrases. The 'ner_tags' column, on the other hand, adheres to the BIO system, labelling abbreviations as 'AC' and expanded forms as 'LF', revealing critical information about how these entities are embedded in the text.

A useful method for visualizing the dataset is to use frequency distribution plots for POS and NER tags to identify prevalent grammatical patterns and named entity occurrences. Furthermore, investigating the co-occurrence of POS and NER tags may reveal complex patterns involving the architecture of abbreviations and long forms within phrases. Analyzing the duration of sequences provides insight into typical sentence forms.

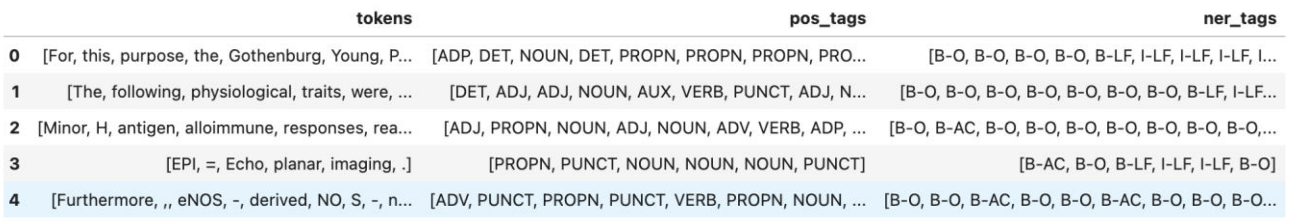
With 1,072 non-null records, the dataset is a solid foundation for training machine learning models. Its low memory utilization indicates efficient processing even on platforms with limited resources.

In conclusion, the initial assessment of the dataset prepares the groundwork for further investigation. Visualizing and interpreting these entries will help to tailor NLP models to the specific properties of scientific writings, ultimately boosting the accuracy of sequence classification tasks that identify abbreviations and extended forms.

2.3 Data Analysis

In this section, we look at the visual and interpretative aspects of the PLOD-CW dataset, which is designed to advance NLP approaches for recognizing scientific abbreviations. Using exploratory data analysis (EDA) approaches and machine learning, we hope to acquire insights into the dataset's features and inform the building of viable models.

After loading the dataset, we use **head()** function For displaying the first few rows of the Data Frame to preview its structure and the types of data it contains.



	tokens	pos_tags	ner_tags
0	[For, this, purpose, the, Gothenburg, Young, P...	[ADP, DET, NOUN, DET, PROPN, PROPN, PROPN, PRO...	[B-O, B-O, B-O, B-O, B-LF, I-LF, I-LF, I-LF, I...
1	[The, following, physiological, traits, were, ...	[DET, ADJ, ADJ, NOUN, AUX, VERB, PUNCT, ADJ, N...	[B-O, B-O, B-O, B-O, B-O, B-O, B-O, B-LF, I-LF...
2	[Minor, H, antigen, alloimmune, responses, rea...	[ADJ, PROPN, NOUN, ADJ, NOUN, ADV, VERB, ADP, ...	[B-O, B-AC, B-O, B-O, B-O, B-O, B-O, B-O, B-O,...
3	[EPI, =, Echo, planar, imaging, .]	[PROPN, PUNCT, NOUN, NOUN, NOUN, PUNCT]	[B-AC, B-O, B-LF, I-LF, I-LF, B-O]
4	[Furthermore, ,, eNOS, -, derived, NO, S, -, n...	[ADV, PUNCT, PROPN, PUNCT, VERB, PROPN, NOUN, ...	[B-O, B-O, B-AC, B-O, B-O, B-AC, B-O, B-O, B-O...

We used **isnull().sum()** function to check if there are an y missing values in the dataset.

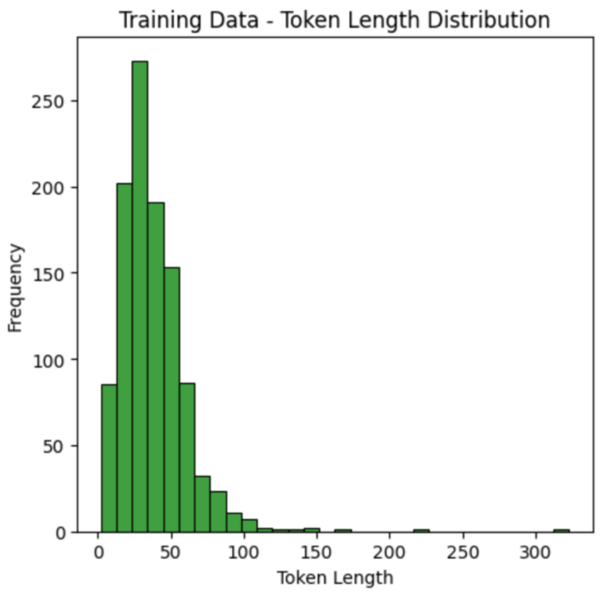
```
Missing values in each column:
tokens      0
pos_tags    0
ner_tags    0
dtype: int64
```

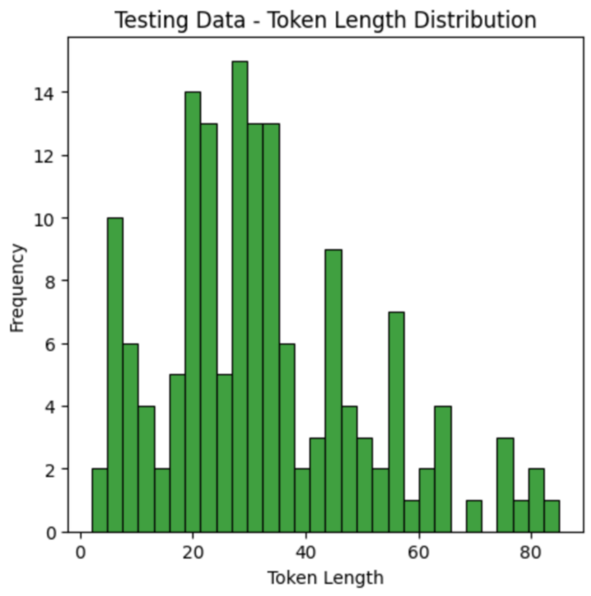
The dataset's visualization and interpretation include a variety of elements, such as the distribution of part-of-speech (POS) tags, named entity recognition (NER) tags, and sequence length. These visualizations assist as first steps towards comprehending the structural and linguistic complexities found in scientific writings.

The frequency distribution plots of POS and NER tags allow us to discover typical grammatical patterns and the prevalence of named entities, notably abbreviations ('AC') and their expanded variants.

Distribution of Token Lengths:

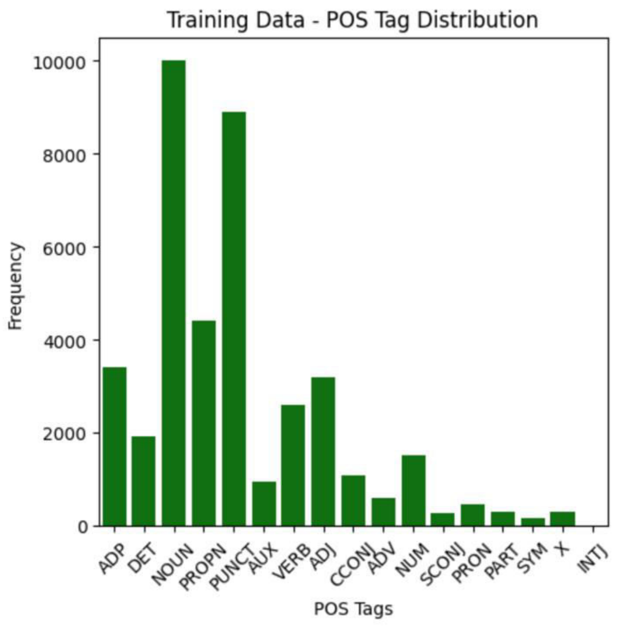
The distribution of token lengths for the training and testing datasets has been represented graphically. These histograms show the variety in the data as well as the typical length of token sequences.

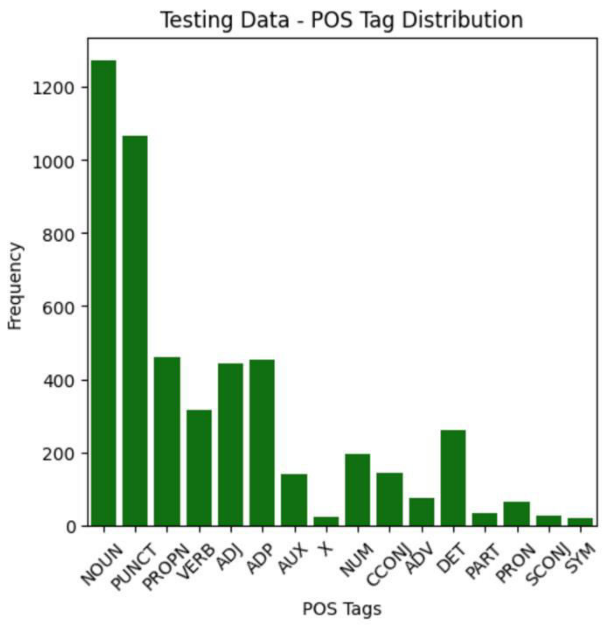




Part-of-Speech (POS) Tag Distribution:

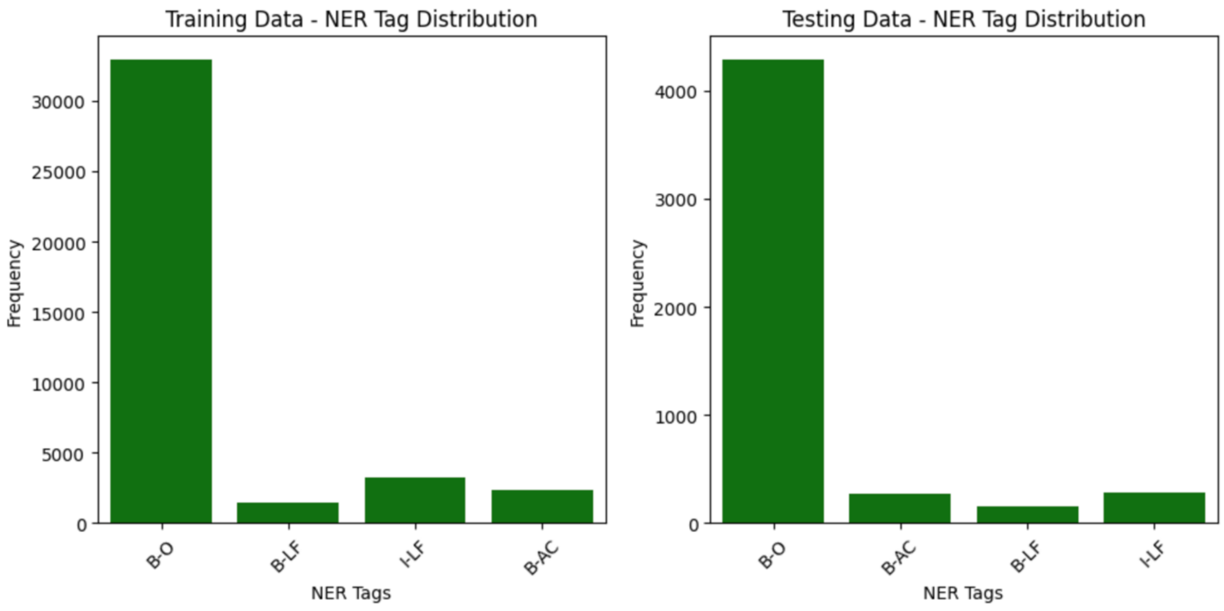
The frequency of POS tags in both data divisions are examined. The count plots show a visual depiction of the dataset's linguistic composition, emphasizing the prevalence of nouns, adjectives and verbs. This gave me insight into the grammatical structures seen in the text data.





Named Entity Recognition (NER) Tag Distribution:

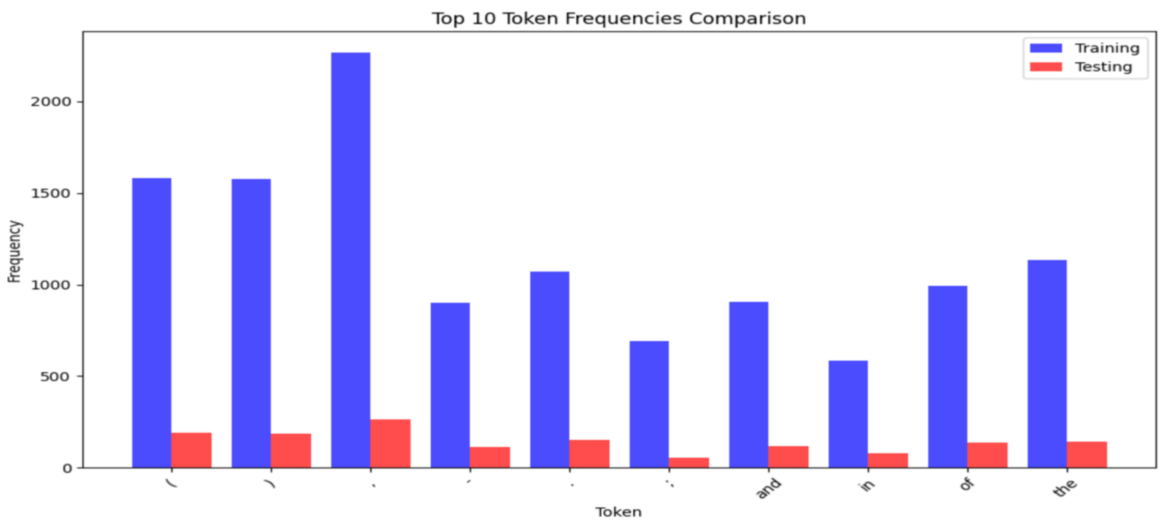
The count graphs show how frequently each sort of identified thing happens. In the training data, I see a dominant frequency of one tag category, indicating a potential imbalance that could affect model training for entity recognition tasks.



These visualizations are crucial for understanding the nature of our dataset and directing future data preparation and model design decisions.

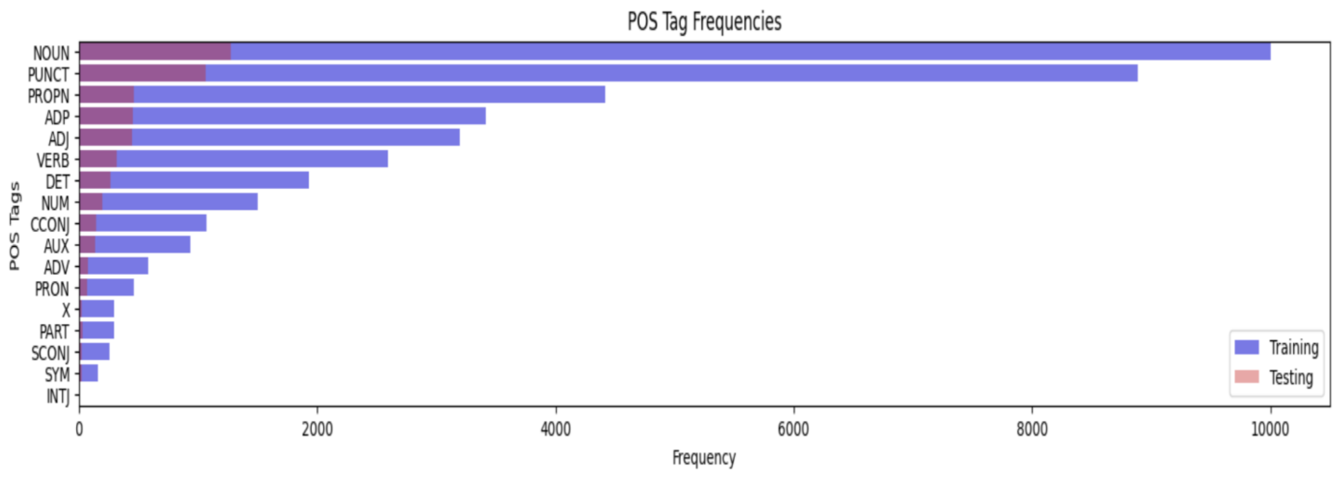
Token Frequency Analysis:

The top ten tokens in both the training and testing datasets are displayed in the first subplot. With the use of this analysis, I was able to determine the most often used words and punctuation, many of which are stop words and popular English words that might not significantly contribute to the text's meaning.



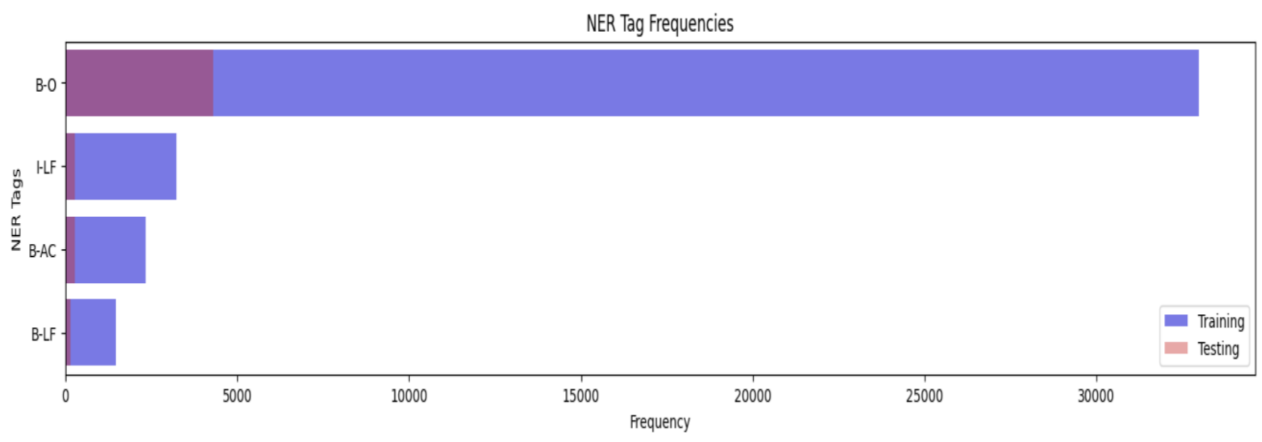
POS Tag Frequency Analysis:

A comparison of each part-of-speech (POS) tag's frequency is shown in the second subplot. This visual analysis facilitates comprehension of the corpus's grammatical structure.



NER Tag Frequency Analysis:

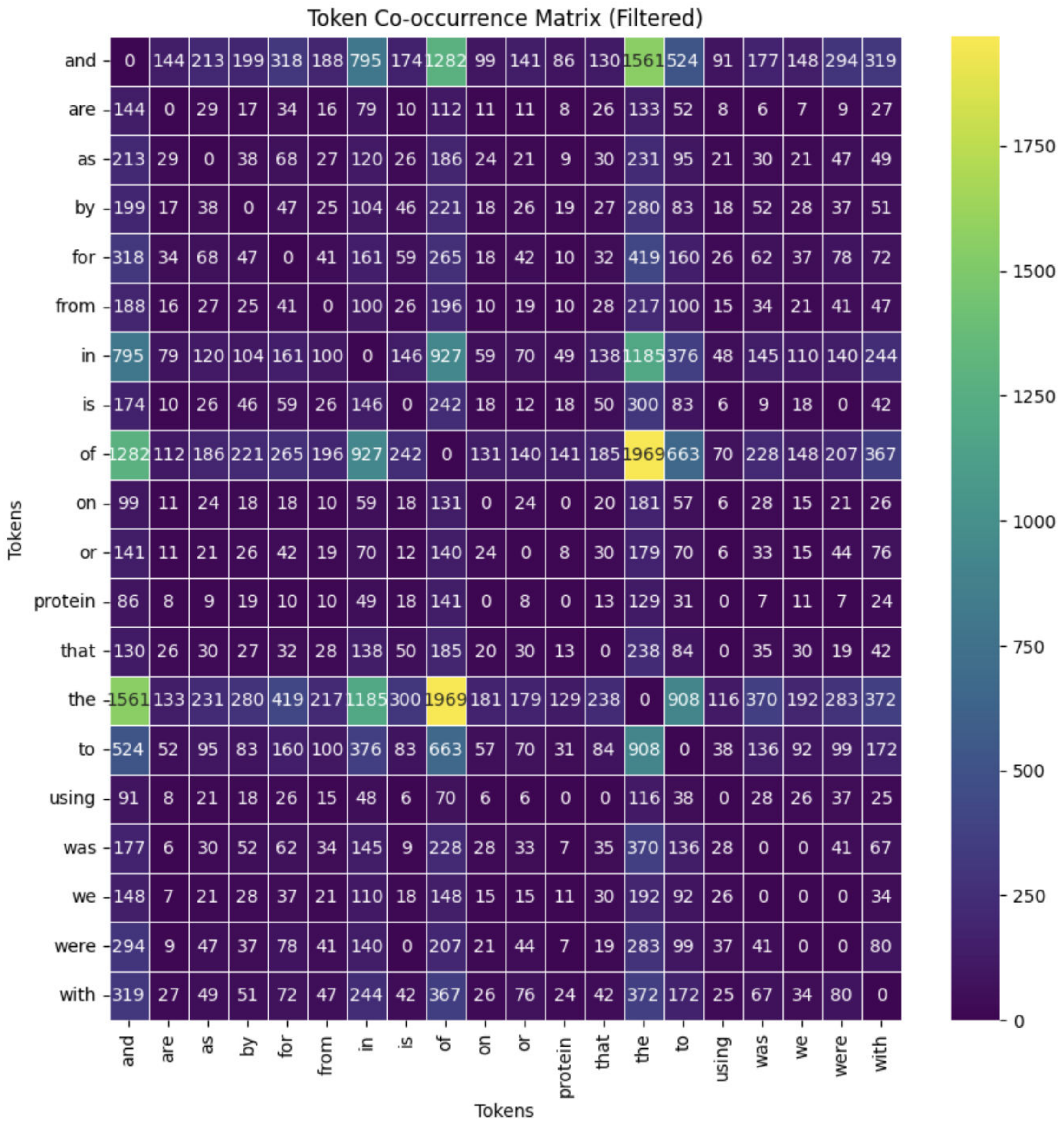
The distribution of entities in the datasets can be accessed with the use of NER tags.



Co-occurrence:

The co-occurrence matrix for tokens in the dataset is shown in the heatmap below. The frequency with which token pairs occur together in the same context is indicated by each cell in the matrix. By highlighting terms that co-occur frequently, this visualisation might provide light on common patterns, themes, or subjects that are present in the dataset.

This visual analysis not only enhances our understanding of the dataset's characteristics but also guides the strategic decisions in the model development process. It is a foundational step that impacts everything from data pre-processing to model evaluation, ultimately influencing the effectiveness and fairness of the predictive models built on this data.



	and	are	as	by	for	from	in	is	of	on	or	protein	that	the	to	using	was	we	were	with
and	0	144	213	199	318	188	795	174	1282	99	141	86	130	1561	524	91	177	148	294	319
are	144	0	29	17	34	16	79	10	112	11	11	8	26	133	52	8	6	7	9	27
as	213	29	0	38	68	27	120	26	186	24	21	9	30	231	95	21	30	21	47	49
by	199	17	38	0	47	25	104	46	221	18	26	19	27	280	83	18	52	28	37	51
for	318	34	68	47	0	41	161	59	265	18	42	10	32	419	160	26	62	37	78	72
from	188	16	27	25	41	0	100	26	196	10	19	10	28	217	100	15	34	21	41	47
in	795	79	120	104	161	100	0	146	927	59	70	49	138	1185	376	48	145	110	140	244
is	174	10	26	46	59	26	146	0	242	18	12	18	50	300	83	6	9	18	0	42
of	1282	112	186	221	265	196	927	242	0	131	140	141	185	1969	663	70	228	148	207	367
on	99	11	24	18	18	10	59	18	131	0	24	0	20	181	57	6	28	15	21	26
or	141	11	21	26	42	19	70	12	140	24	0	8	30	179	70	6	33	15	44	76
protein	86	8	9	19	10	10	49	18	141	0	8	0	13	129	31	0	7	11	7	24
that	130	26	30	27	32	28	138	50	185	20	30	13	0	238	84	0	35	30	19	42
the	1561	133	231	280	419	217	1185	300	1969	181	179	129	238	0	908	116	370	192	283	372
to	524	52	95	83	160	100	376	83	663	57	70	31	84	908	0	38	136	92	99	172
using	91	8	21	18	26	15	48	6	70	6	6	0	0	116	38	0	28	26	37	25
was	177	6	30	52	62	34	145	9	228	28	33	7	35	370	136	28	0	0	41	67
we	148	7	21	28	37	21	110	18	148	15	15	11	30	192	92	26	0	0	0	34
were	294	9	47	37	78	41	140	0	207	21	44	7	19	283	99	37	41	0	0	80
with	319	27	49	51	72	47	244	42	367	26	76	24	42	372	172	25	67	34	80	0

3. Experimentation

3.1 Experiment 1: Vectorization techniques and 1D Convolutional Neural Network (1D CNN) comparison

Objective:

The objective is to determine which vectorization technique is more effective for pre-processing text data and improving the performance of 1D CNN models in sequence classification tasks.

By comparing the findings of TF-IDF and count vectorization, we may learn about how these algorithms treat text data differently and whether they are appropriate for different forms of textual information. Furthermore, this comparison will assist in determining the strengths and shortcomings of each technique in terms of computational efficiency, scalability, and ease of implementation.

Methodology:

- **Combine Tokens:** Combining individual tokens into single strings for each sample in the training and test datasets.
- **Vectorization (TF-IDF):** TfidfVectorizer is used to convert the train and test texts into a matrix of TF-IDF features.
- **Vectorization (Count):** CountVectorizer is also used to transform texts into a matrix of token counts. This creates a different representation that could be used to train models as an alternative to TF-IDF features.
- **Label Encoding:** Creating a LabelEncoder instance, which is used to encode the labels numerically. This is necessary because machine learning models don't work with categorical labels directly; they require numerical inputs.
- **One-Hot Encoding of Labels:** The numerical labels are then converted to a one-hot encoded format using to categorical, which is necessary for classification with neural networks since the output layer will likely use SoftMax activation that expects this format. This step creates a binary matrix representation of the labels.
- **Building the 1D CNN Model:** A 1D CNN model is created using a stack of layers. It starts with a convolutional layer that filters the input text, followed by a pooling layer that simplifies the output from the convolutional layer, and ends with a dense layer that decides the class of the input text based on the features extracted by the convolutional layer.
- **Training the Model with TF-IDF Data:** The model is then taught using the TF-IDF transformed text data. TF-IDF is a way to turn text into numbers that the model can understand, highlighting which words are important in a document.
- **Making Predictions with TF-IDF Data:** After learning, the model uses what it knows to make predictions on new, unseen text data that has also been turned into numbers with TF-IDF.
- **Training the Model with Count Vector Data:** Now The model learns from count vector data, which is another way to turn text into numbers by simply counting how often each word appears.
- **Making Predictions with Count Vector Data:** The model, now trained with count vector data, makes another set of predictions on new, unseen text data that has been turned into numbers using the count method.
- **Evaluating the Models:** The model's performance is checked by comparing its predictions to the actual classes of the text data. A report is printed showing how well the model did for each type of class it was supposed to identify.
- **Visualizing the Results:** The results are also shown in confusion matrices, which are special charts that help visualize where the model is making correct predictions and where it's getting confused. These charts show the predicted classes against the actual classes using colour intensity to indicate the number of predictions.

3.2 Experiment 2: Comparing 1D CNN and BiLSTM models using TF-IDF Vectorizer

Objective:

The objective of comparing 1D CNN and BiLSTM models using TF-IDF vectorization is to evaluate the performance of these two deep learning architectures in sequence classification tasks. The comparison involves assessing key performance metrics such as accuracy, precision, recall, and F1-score on a common dataset. By analyzing these metrics, we can determine which model architecture, either 1D CNN or BiLSTM, is better suited for the given task when combined with TF-IDF vectorization.

Methodology:

- **Preparing Text Data:** The text data from the loaded dataset is prepared by concatenating the tokens for each sample into a single string, using list comprehensions and the join function.
- **TF-IDF Vectorization:** The TfidfVectorizer is used to convert the text data into TF-IDF vectors. The fit transform method is called on the training data to fit the vectorizer and transform the training texts, while the transform method is used on the test data to transform it based on the vectorizer fitted with the training data.
- **Encoding Labels:** The labels from the dataset's NER tags are encoded using LabelEncoder. This step assigns a unique integer to each class in the labels, making it suitable for model training.
- **Converting Labels to One-Hot Encoding:** The encoded labels are then converted into one-hot encoded format using the to categorical function. This is necessary for multi-class classification tasks as it represents each class as a binary vector.
- **Building and Training 1D CNN Model:** A sequential model (model CNN) is created with a convolutional layer (Conv1D), global max pooling layer (GlobalMaxPooling1D), and a dense layer with SoftMax activation for multi-class classification. The model is compiled with the Adam optimizer and categorical crossentropy loss function. The model is trained (history CNN) on the TF-IDF transformed training data for 10 epochs with a validation split of 20% and a batch size of 32.
- **Making Predictions with 1D CNN Model:** Predictions (y_pred_cnn) are made on the TF-IDF transformed test data using the trained 1D CNN model.
- **Building and Training BiLSTM Model:** Another sequential model (model_bilstm) is created with a bidirectional LSTM layer (Bidirectional), global max pooling layer, and dense layer with SoftMax activation. Similar to the 1D CNN model, the BiLSTM model is compiled and trained (history_bilstm) on the TF-IDF transformed training data for 10 epochs with validation split and batch size.
- **Making Predictions with BiLSTM Model:** Predictions (y_pred_bilstm) are made on the TF-IDF transformed test data using the trained BiLSTM model.
- **Printing Classification Reports:** Classification reports are printed for both models, showing precision, recall, and F1-score for each class in the test data. These reports provide insights into the models' performance on different classes and overall accuracy.
- **Visualizing Confusion Matrices:** Confusion matrices are plotted for both models, providing a visual representation of the models' predictions compared to the actual labels. The heatmaps in the confusion matrices help in understanding where the models are making correct predictions and where they are getting confused.

3.3 Experiment 3: Comparing Grid search and Random search techniques for 1D CNN Model

Objective:

The objective of this comparison is to evaluate the efficacy and efficiency of two popular hyperparameter tuning techniques, Grid Search and Random Search, in optimizing the performance of a 1D CNN (Convolutional Neural Network) model. We will evaluate the classification performance metrics (e.g., accuracy, precision, recall, F1-score) achieved by models tuned using Grid Search and Random Search. This assessment will help identify which technique leads to better-performing models.

Methodology:

- **Data Preparation:** Text data from the dataset is pre-processed, and TF-IDF vectorization is performed to convert text data into numerical features. Labels are encoded using LabelEncoder and transformed into one-hot encoded format for model training.
- **Model Definition:** A function create_cnn_model is defined to construct a 1D CNN model with configurable hyperparameters. This model consists of a convolutional layer, global max pooling layer, and a dense layer with softmax activation for classification. The model is compiled with categorical crossentropy loss and configured optimizer and metrics.
- **Hyperparameter Tuning Setup:** KerasClassifier from scikit-learn is configured to work with the defined CNN model for use in hyperparameter tuning. A parameter grid is specified with different combinations of hyperparameters such as filters, kernel size, activation function, batch size, and number of epochs.
- **Grid Search and Randomized Search:** Grid Search (GridSearchCV) and Randomized Search (RandomizedSearchCV) are set up with the specified parameter grid and configured to evaluate model performance using accuracy as the scoring metric. Grid Search exhaustively evaluates all possible combinations of hyperparameters, while Randomized Search randomly samples a specified number of combinations.
- **Model Fitting:** Both Grid Search and Randomized Search are applied to fit the models on the training data, searching for the best hyperparameters that optimize model performance.
- **Evaluation:** The best hyperparameters found by both search methods are outputted. Confusion matrices are computed to evaluate the performance of the models on the test data, visualizing the true positive, false positive, true negative, and false negative predictions for each class.
- **Visualization:** Confusion matrices are plotted using seaborn and matplotlib to provide a visual representation of model performance, helping to identify areas of correct and incorrect predictions for each class.

3.4 Experiment 4: Comparing Categorical Crossentropy and Sparse Categorical Crossentropy for 1D CNN model

Objective:

The objective is to evaluate and compare the performance of Categorical Crossentropy and Sparse Categorical Crossentropy loss functions when training a 1D Convolutional Neural Network (CNN) for classification tasks. This analysis will look specifically at how these loss functions affect model correctness, training stability, and computational efficiency. This comparison will aid in determining the best loss function to improve the efficacy of 1D CNN models in handling categorical data in a variety of classification tasks.

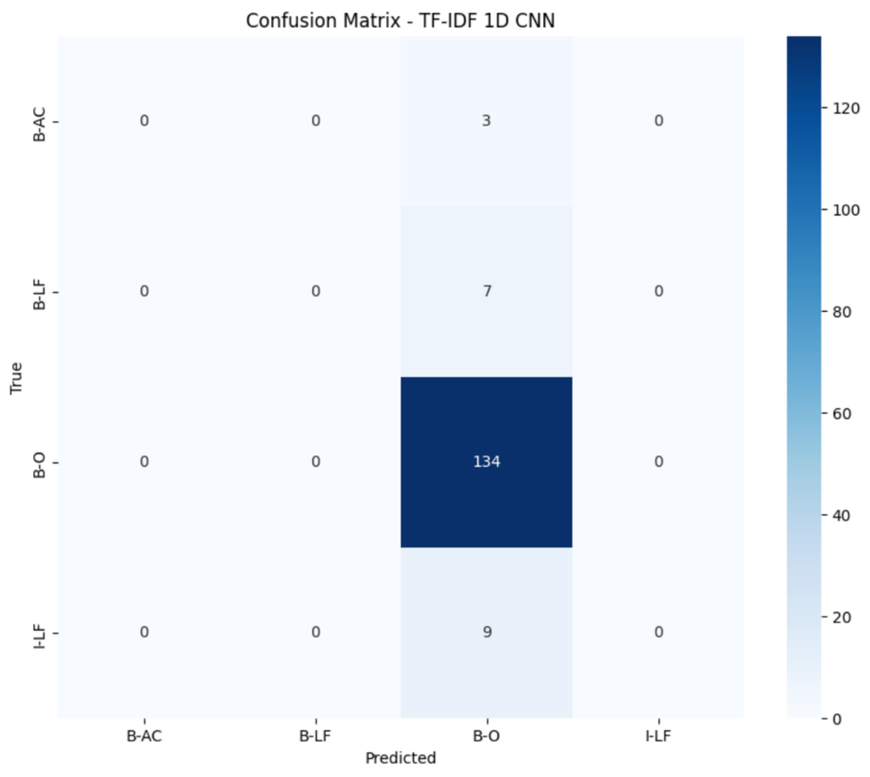
Methodology:

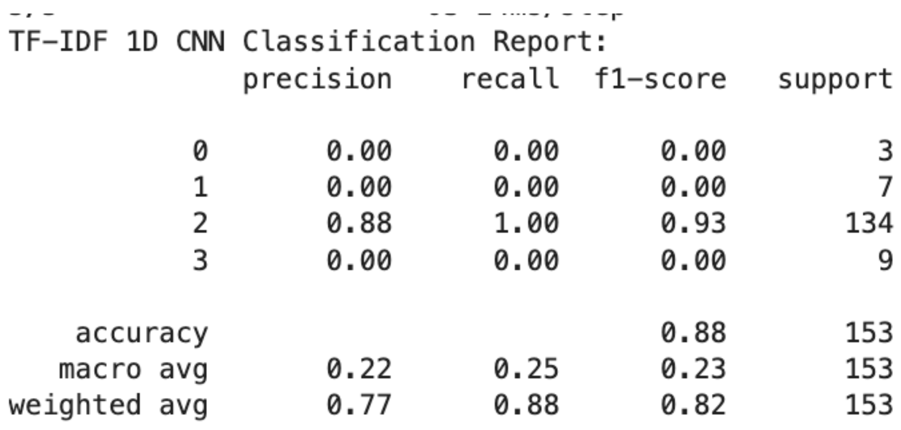
- **Data Preparation:** Text data from the dataset is vectorized using TF-IDF, and labels are encoded and transformed into one-hot encoding.
- **Model Construction:** Two separate 1D CNN models are built, each using a different loss function: Categorical Crossentropy and Sparse Categorical Crossentropy.
- **Training:** Both models are trained on the TF-IDF vectorized data with identical hyperparameters and training procedures.
- **Evaluation:** The performance of each model is evaluated using standard metrics such as accuracy, precision, recall, and F1-score on a separate test dataset.
- **Visualization:** The training history (loss and accuracy) of each model is plotted to observe the training process and convergence behaviour.
- **Analysis:** Classification reports and confusion matrices are generated to analyse the performance of each model and compare their effectiveness in classifying text data.

4. Analyzing Experimentation Results

4.1 Analyzing Experiment 1:

TF-IDF Vectorization with 1D CNN





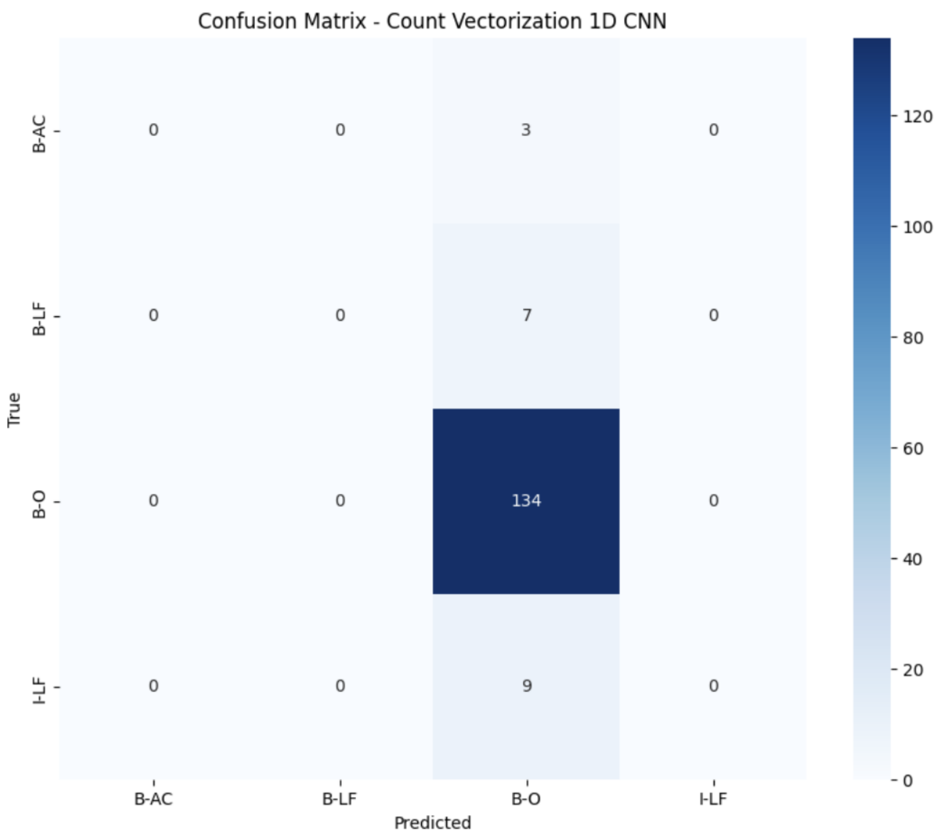
```
TF-IDF 1D CNN Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         7
           2       0.88      1.00      0.93       134
           3       0.00      0.00      0.00         9

    accuracy                           0.88       153
   macro avg       0.22      0.25      0.23       153
weighted avg       0.77      0.88      0.82       153
```

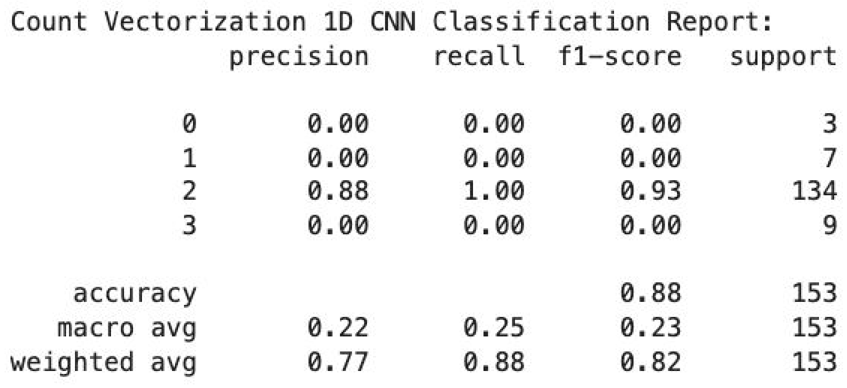
The classification report and confusion matrix above show the performance of a 1D CNN model with TF-IDF (Term Frequency-Inverse Document Frequency) vectorization. The model was charged with categorising samples into four different groups, identified as classes 0, 1, 2, and 3.

Classification Performance:

- Precision, Recall, and F1-Score: The model performs well in class 2 with an F1-score of 0.93, indicating effective classification with a precision of 0.88 and a recall of 1.00. This implies that the model accurately predicts class 2 88% of the time and correctly identifies all actual cases of class 2. However, the model fails to predict classes 0, 1, and 3 completely, as indicated by zero scores across precision, recall, and F1-score.
- Accuracy: The overall accuracy of the model is 0.88, which is misleadingly high due to the imbalanced dataset where most samples belong to class 2.
- Macro and Weighted Averages: The macro average F1-score of 0.23 and weighted average F1-score of 0.82 reveal a stark difference in performance across classes. The macro average treats all classes equally, highlighting the poor performance for classes 0, 1, and 3, while the weighted average is skewed by the high number of instances in class 2.
- Confusion Matrix Insights: The confusion matrix reinforces the above observations, where all predictions are made for class 2, with 134 correct predictions and the rest being false negatives for other classes.

<u>Count Vectorization with 1D CNN</u>





```
Count Vectorization 1D CNN Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         7
           2       0.88      1.00      0.93       134
           3       0.00      0.00      0.00         9

    accuracy                           0.88       153
   macro avg       0.22      0.25      0.23       153
weighted avg       0.77      0.88      0.82       153
```

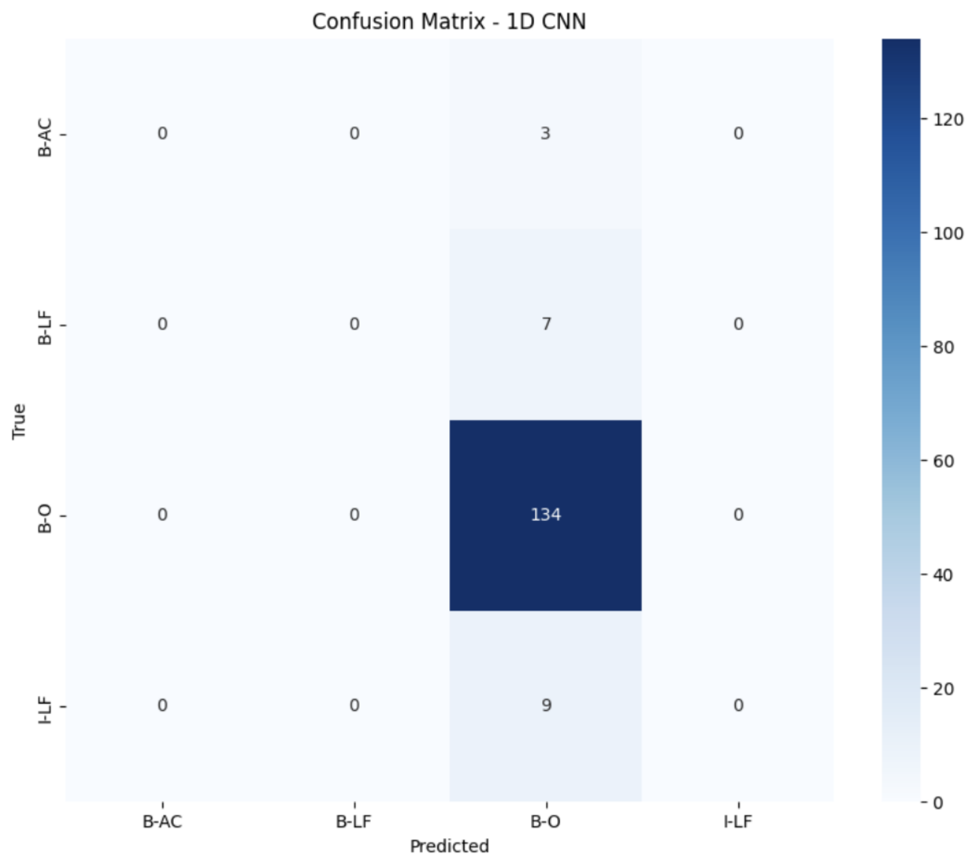
The Analysis illustrates the performance outcomes of a 1D CNN model trained using Count Vectorization for text classification. The evaluation is based on a classification report and a confusion matrix, which together offer insight into the model's predictive capabilities across four classes labelled as 0, 1, 2, and 3.

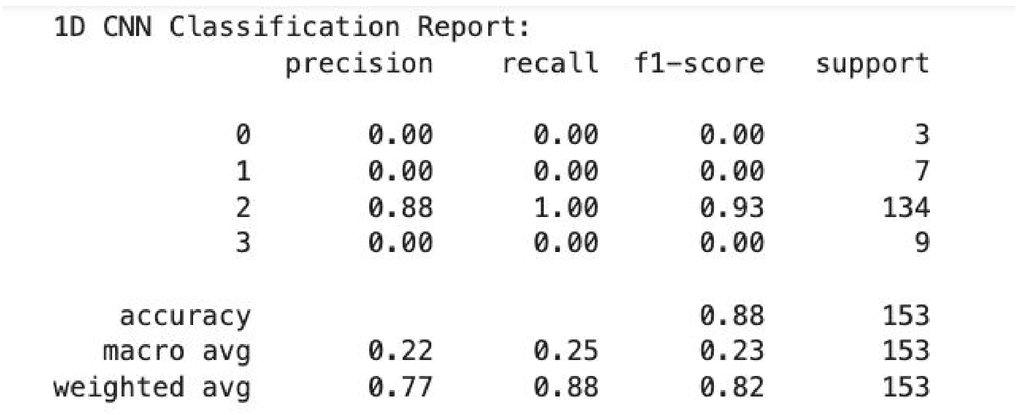
Classification Performance:

- Precision, Recall, and F1-Score: There is a striking disparity in the model’s ability to predict different classes. Classes 0, 1, and 3 have precision, recall, and F1-scores of 0.00, indicating the model failed to correctly predict any samples from these categories. Class 2, in contrast, shows a high F1-score of 0.93, with precision at 0.88 and recall at 1.00, denoting that while the model reliably identifies class 2 instances, it tends to incorrectly label other classes as class 2.
- Accuracy: The model achieves an overall accuracy of 0.88, which is heavily skewed by the high prevalence of class 2 in the dataset. Such a high accuracy may not reflect true predictive performance across all classes due to this imbalance.
- Macro and Weighted Averages: The macro average scores (0.22 for precision, 0.25 for recall, and 0.23 for F1-score) show poor performance when all classes are treated equally. The weighted average, however, is significantly higher (0.77 for precision, 0.88 for recall, and 0.82 for F1-score), reflecting the over-representation of class 2 in the dataset and the model's proficiency in predicting this class.
- Confusion Matrix Observations: The matrix confirms that all samples from classes 0 and 3, along with most from class 1, were misclassified as class 2. It’s evident that class 2 is the dominant class, with 134 correct predictions and no misclassifications into other classes. Classes 0 and 3 are the most negatively impacted, with no true positive predictions.

4.2 Analyzing Experiment 2:

1D CNN With TF-IDF Vectorizer





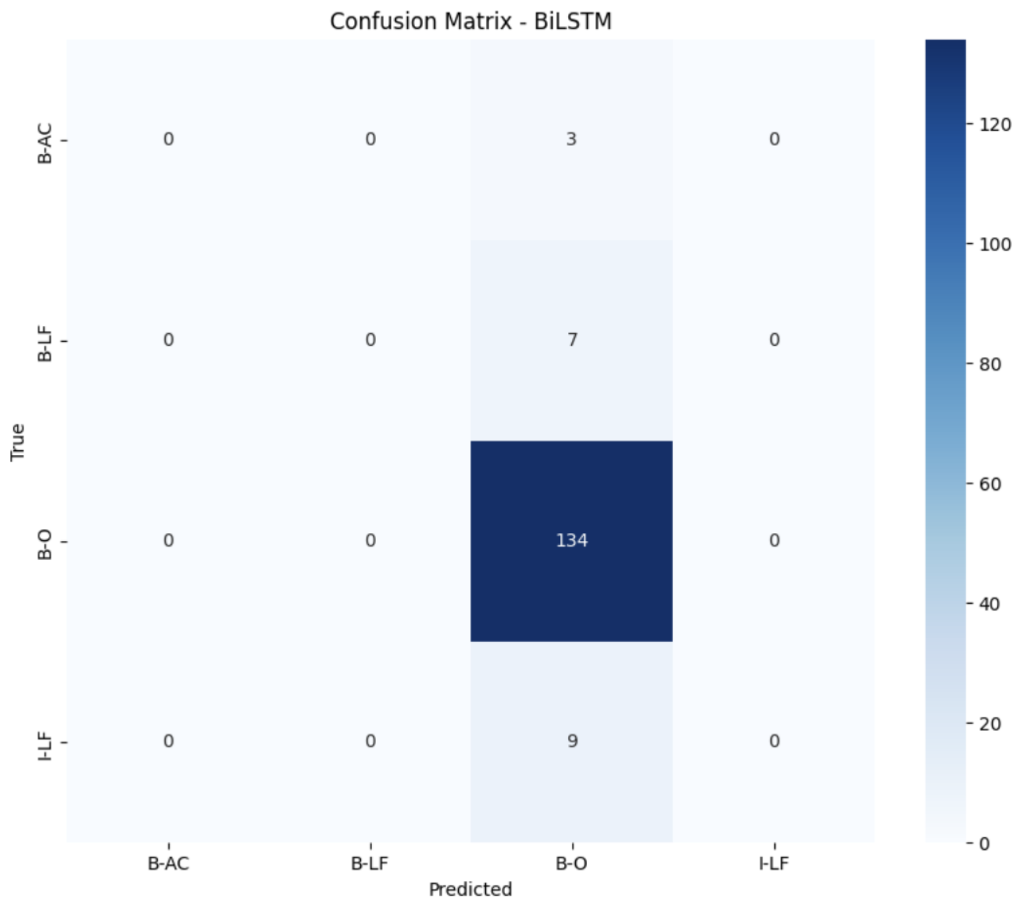
```
1D CNN Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         7
           2       0.88      1.00      0.93       134
           3       0.00      0.00      0.00         9

    accuracy                           0.88       153
   macro avg       0.22      0.25      0.23       153
weighted avg       0.77      0.88      0.82       153
```

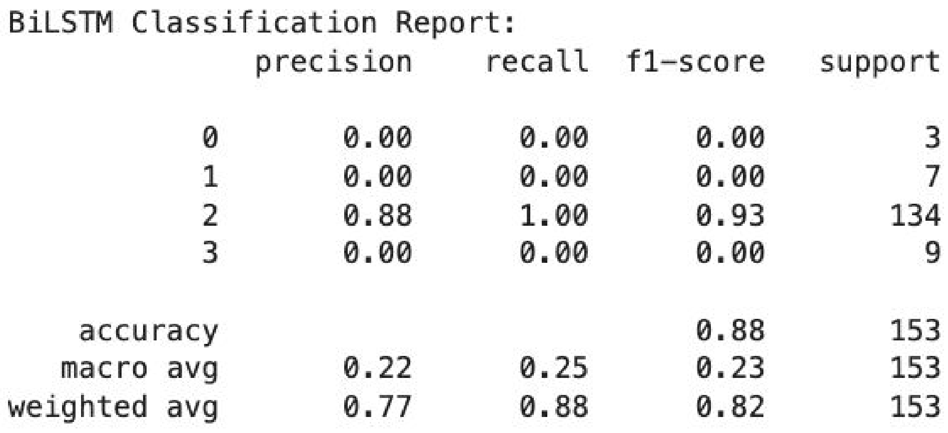
This Analysis provides an evaluation of the 1D Convolutional Neural Network (CNN) for a classification task, based on the provided classification report and confusion matrix.

Classification Performance:

- Precision and Recall: The precision and recall for classes 0, 1, and 3 are zero, indicating that the model did not correctly predict any true positives for these classes. This suggests that the model's capability to generalize across various classes is lacking, and it is particularly unable to identify features that are indicative of these classes.
- F1-Score: The F1-score, which balances precision and recall, is also zero for classes 0, 1, and 3, underscoring the model's inability to predict these classes correctly. For class 2, the F1-score is 0.93, demonstrating a high degree of precision and recall. However, this score is not representative of the overall model's performance across all classes.
- Overall Accuracy: An overall accuracy of 0.88 is observed, which is disproportionately influenced by the model's performance on class 2, the majority class.
- Macro and Weighted Averages: The macro average scores are low (0.22 for precision, 0.25 for recall, and 0.23 for F1-score), reflecting the poor performance across all classes when treated equally. The weighted averages are higher due to the influence of the dominant class 2, but they mask the model's deficiencies in classifying other classes.
- Confusion Matrix Insights: Misclassifications: The confusion matrix visualizes the misclassification pattern where all instances of classes 0 and 3, and most from class 1, are incorrectly predicted as class 2.
- Class 2 Overprediction: The model exhibits a strong bias towards predicting class 2, likely due to its over-representation in the training data.

<u>**BiLSTM With TF-IDF Vectorizer**</u>





```
BiLSTM Classification Report:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         7
           2       0.88      1.00      0.93       134
           3       0.00      0.00      0.00         9

    accuracy                           0.88       153
   macro avg       0.22      0.25      0.23       153
weighted avg       0.77      0.88      0.82       153
```

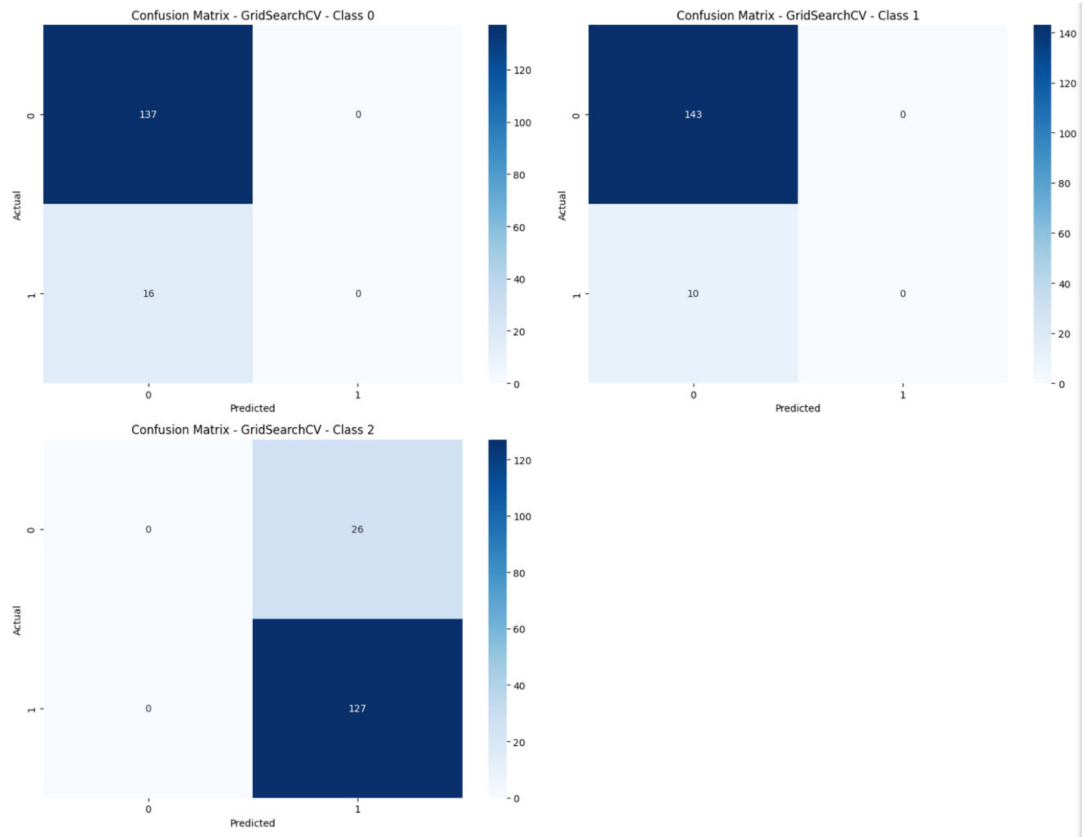
- This depicts a detailed analysis of a Bidirectional Long Short-Term Memory (BiLSTM) network's performance on a classification task, as depicted in the provided classification report and confusion matrix.

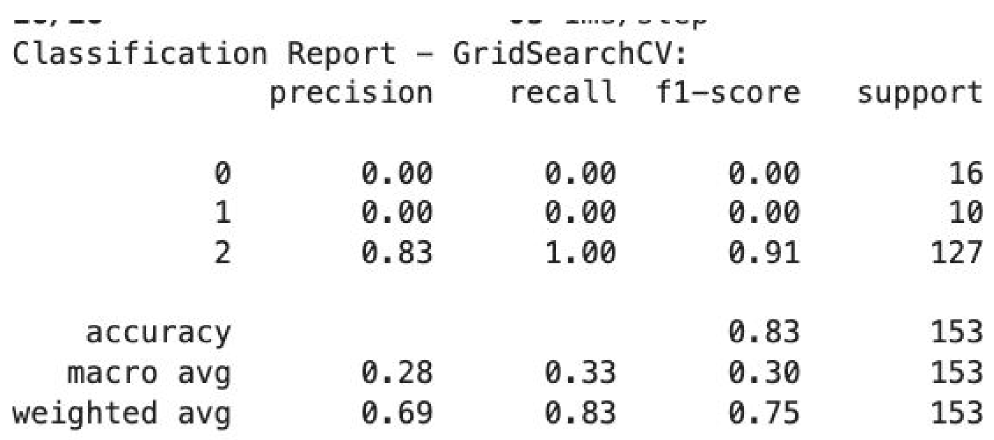
Classification Report Analysis:

- Precision, Recall, and F1-Score: The model has completely failed to identify any samples correctly in classes 0, 1, and 3, as evidenced by the scores of 0.00 across precision, recall, and F1-score. Class 2 stands out with a precision of 0.88 and a recall of 1.00, resulting in a high F1-score of 0.93. This indicates the model’s proficiency in classifying this specific class, but it also underscores its inability to distinguish the other classes.
- Accuracy: The overall model accuracy is 0.88, but this metric is not representative of the model's performance across all classes due to the disproportionate representation of class 2 in the dataset.
- Macro and Weighted Averages: The macro average scores (precision: 0.22, recall: 0.25, F1-score: 0.23) reflect poor model performance when evaluating all classes equally. The weighted averages (precision: 0.77, recall: 0.88, F1-score: 0.82) are inflated due to the high number of class 2 instances.
- Confusion Matrix Insights: Class Imbalance Impact: The confusion matrix visualizes a clear bias in predictions towards class 2. All instances of classes 0 and 3, and most from class 1, are erroneously classified as class 2, showing the model’s tendency to favor the majority class.
- No Cross-Class Predictions: There are no instances where classes 0, 1, or 3 are predicted as anything other than class 2, indicating a significant limitation in the model’s ability to differentiate between classes.

4.3 Analyzing Experiment 3:

GridSearch With 1D CNN





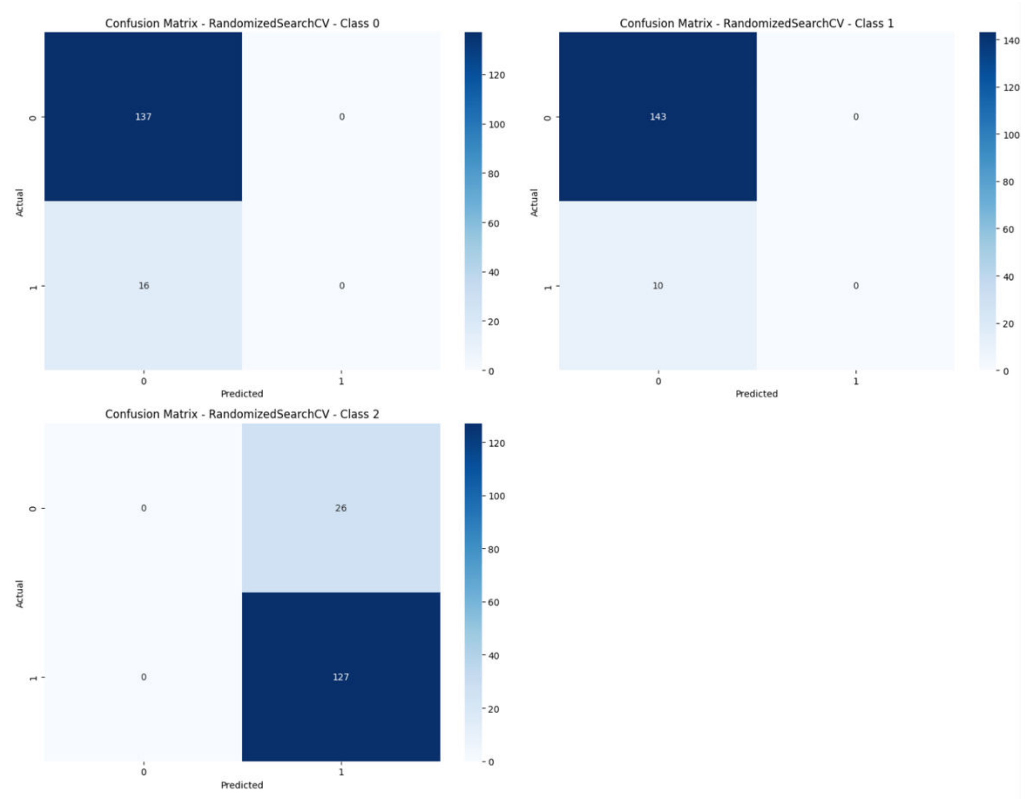
```
Classification Report - GridSearchCV:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.00      0.00      0.00        10
           2       0.83      1.00      0.91       127

    accuracy                           0.83       153
   macro avg       0.28      0.33      0.30       153
weighted avg       0.69      0.83      0.75       153
```

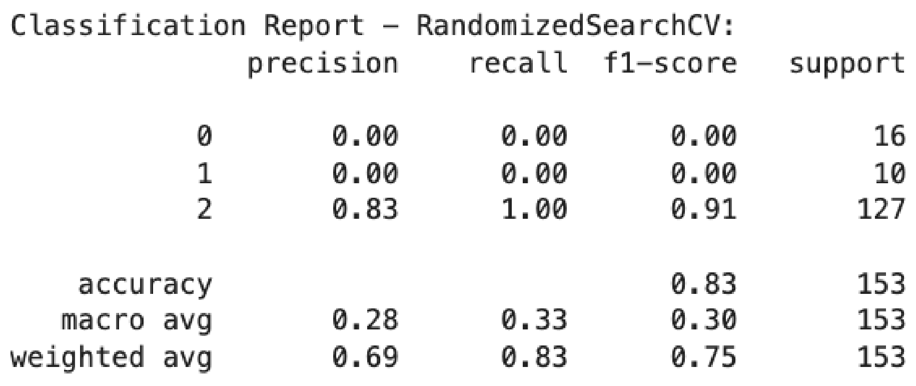
Based on the confusion matrices and the classification report for a model optimized using GridSearchCV, the following observations can be included in the report:

Performance Overview:

- The model demonstrates a strong predictive ability for one class (Class 2), with an F1-score of 0.91. This is supported by high precision (0.83) and perfect recall (1.00), indicating that when the model predicts an instance as belonging to Class 2, it is very likely to be correct, and it successfully captures all instances of Class 2 in the dataset.
- Conversely, the model completely fails to identify instances of Classes 0 and 1, as indicated by the precision, recall, and F1-scores of 0.00 for both classes. This suggests that the features used by the model are not sufficient to distinguish between these classes, or the model may have overfitted to Class 2.
- Confusion Matrix Analysis: For Class 0, the model incorrectly predicts 16 instances as belonging to Class 1, with no correct predictions (true positives).
- In the case of Class 1, the model again predicts all instances incorrectly as Class 0 (10 misclassifications).
- Class 2 is where the model excels, correctly identifying 127 instances and incorrectly classifying 26 instances as Class 0.
- Accuracy and Averages: The overall accuracy of the model is 0.83, which is relatively high but not indicative of a well-rounded performance across all classes. This is a clear example of accuracy being a misleading metric in the face of class imbalance.
- The macro averages (precision: 0.28, recall: 0.33, F1-score: 0.30) are significantly lower than the weighted averages (precision: 0.69, recall: 0.83, F1-score: 0.75), reflecting the skewness of the data towards Class 2.

<u>**RandomSearch With 1D CNN**</u>





```
Classification Report - RandomizedSearchCV:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00        16
           1       0.00      0.00      0.00        10
           2       0.83      1.00      0.91       127

    accuracy                           0.83       153
   macro avg       0.28      0.33      0.30       153
weighted avg       0.69      0.83      0.75       153
```

The confusion matrices and classification report provide valuable insights into the performance of a model optimized with RandomizedSearchCV. Here's an analytical summary that could be included in a report:

Class-specific Performance:

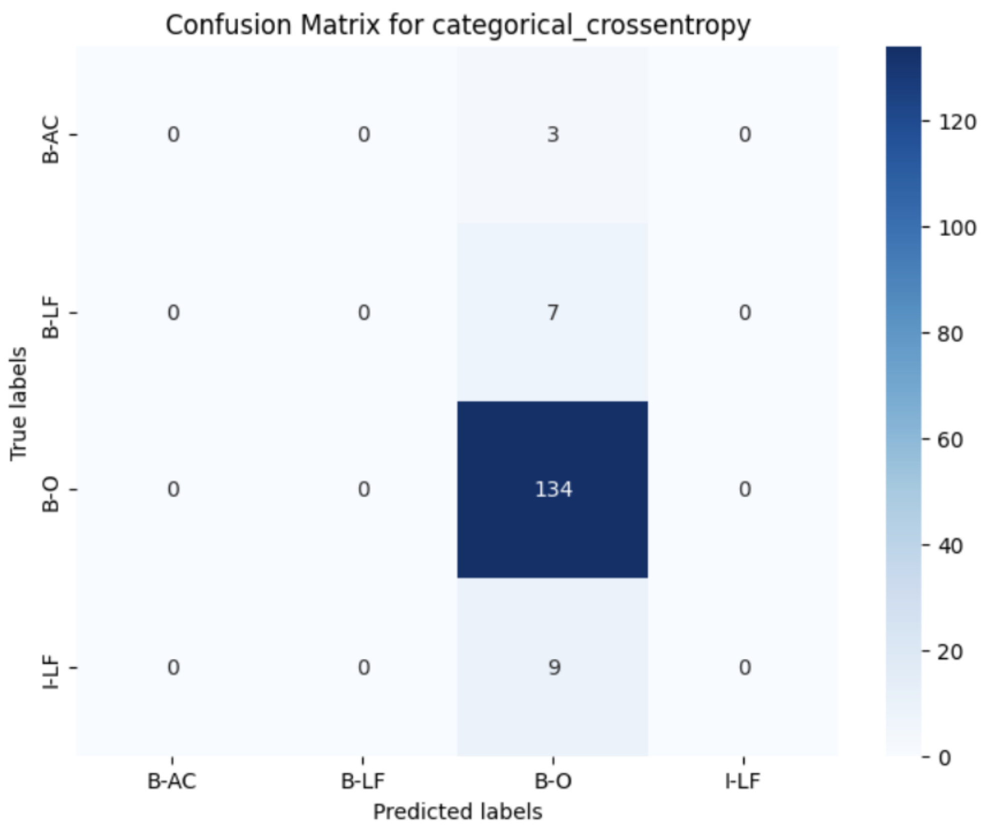
- Class 0: The model has a high true negative rate, correctly identifying 137 instances as not belonging to Class 0. However, it fails to correctly identify any true positives, as evidenced by 16 false negatives.
- Class 1: Similarly to Class 0, the model successfully identifies 143 instances as not belonging to Class 1, yet it incorrectly classifies all actual Class 1 instances (10) as belonging to Class 0.
- Class 2: The model excels in identifying Class 2 instances, with 127 true positives and only 26 false negatives. This indicates the model has learned the features for Class 2 very well.

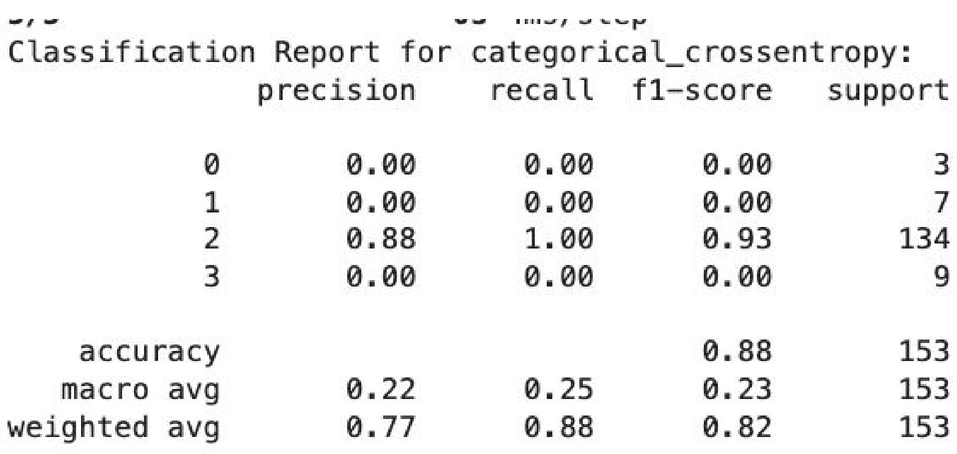
Overall Metrics:

- The model's precision for Class 2 is strong (0.83), indicating that most instances predicted as Class 2 are indeed Class 2. However, the absence of precision for Classes 0 and 1 indicates the model never predicts these classes, which is a serious concern.
- The recall for Class 2 is perfect (1.00), which means every instance of Class 2 in the dataset is captured. Yet again, the recall for Classes 0 and 1 is 0.00, suggesting that the model fails to correctly identify any true instances of these classes.
- The F1-score for Class 2 is high (0.91), a harmonic mean of precision and recall, signifying effective identification of this class. The F1-scores for Classes 0 and 1 are 0.00, highlighting a complete misclassification for these classes.
- Accuracy is reported as 0.83, which is significantly high, but this is mainly because the model performs well in predicting the majority class (Class 2). This metric does not reflect the inability to predict Classes 0 and 1.
- The macro average F1-score (0.30) is quite low, indicating poor performance across all classes when each class is given equal importance. The weighted average F1-score (0.75) is higher due to the high number of Class 2 instances, skewing the average towards the model's strength.

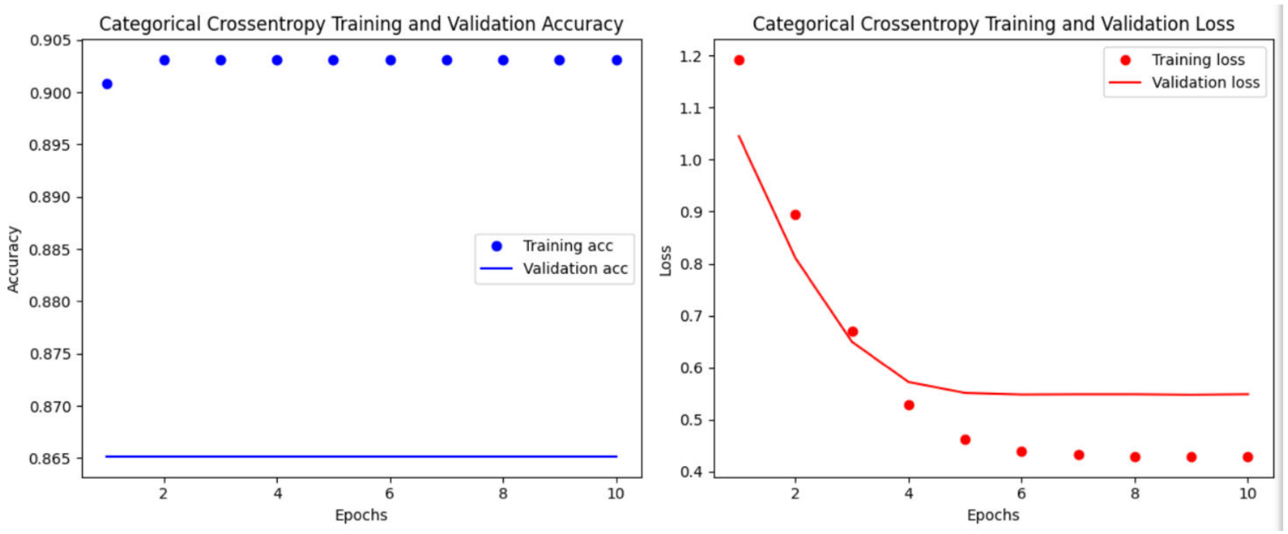
4.4 Analyzing Experiment 4:

Categorical Crossentropy With 1D CNN





```
Classification Report for categorical_crossentropy:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         7
           2       0.88      1.00      0.93       134
           3       0.00      0.00      0.00         9

    accuracy                           0.88       153
   macro avg       0.22      0.25      0.23       153
weighted avg       0.77      0.88      0.82       153
```

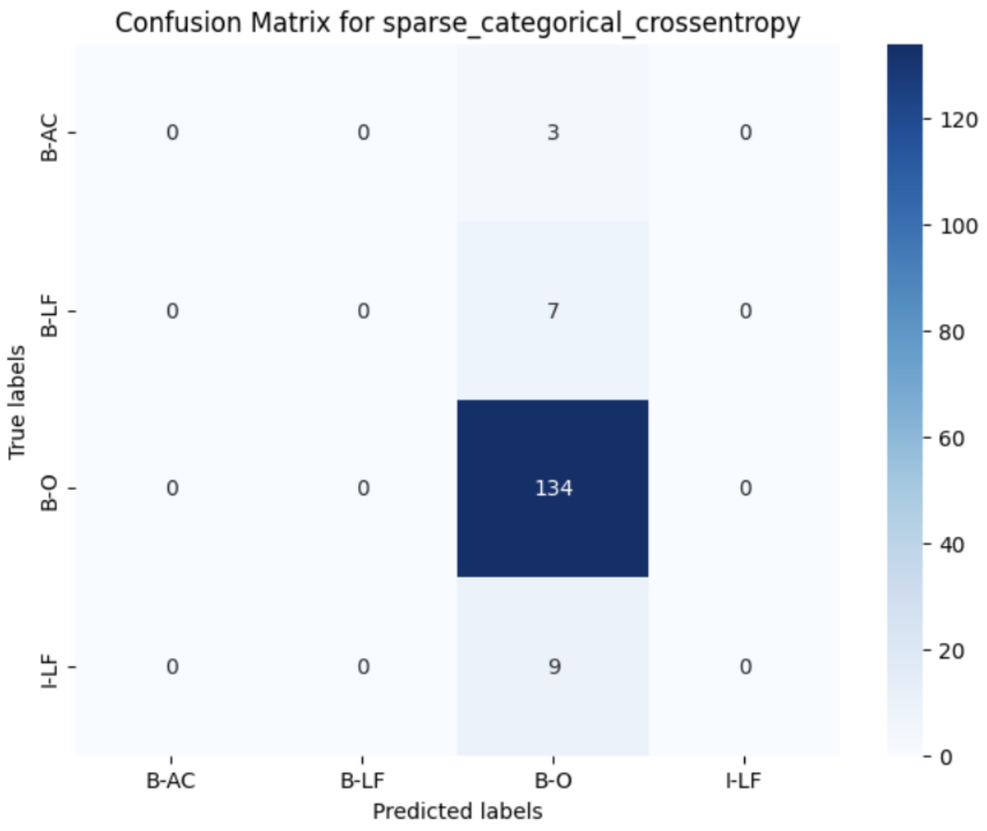


Based on the provided confusion matrix, classification report, and training/validation accuracy and loss charts for a model trained with categorical crossentropy, the following observations and analyses were made.

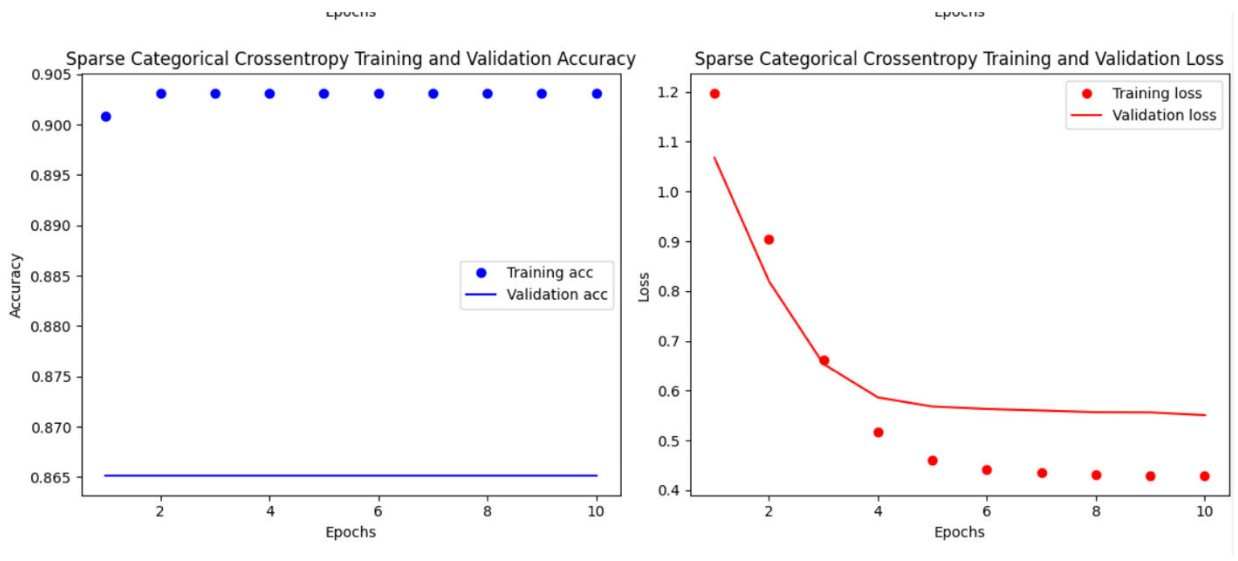
Performance Overview:

- The confusion matrix reveals a concentrated accuracy in predicting Class 2, with 134 true positives and no false positives or false negatives. This indicates a robust identification for this class.
- Classes 0, 1, and 3, however, show a worrying trend, with no true positives at all. Class 0 and Class 3 have been incorrectly classified as Class 2 in all cases, while the confusion matrix for Class 1 is not provided, its classification metrics suggest a similar trend.
- The classification report corroborates this observation, with an F1-score of 0.93 for Class 2, reflecting high precision and perfect recall. Conversely, Classes 0, 1, and 3 have F1-scores of 0.00, signaling complete misclassification.
- Overall model accuracy is high at 0.88, but this figure is misleading as it primarily reflects the successful prediction of the over-represented Class 2.
- Macro averages across precision, recall, and F1-score are low (all below 0.25), indicating poor performance across classes when each class is equally weighted.
- Weighted averages are higher due to the influence of the correctly predicted majority class but do not reflect the model's performance on minority classes.
- Training and Validation Accuracy and Loss Analysis: The training and validation accuracy graphs show a plateau, with both measures reaching stability early in the training process. This could indicate a model that quickly learned to classify the predominant class but failed to generalize across all classes.
- Training and validation loss graphs show a clear convergence, with training loss decreasing smoothly and validation loss stabilizing, suggesting that the model is not overfitting.

Sparse Categorical Crossentropy With 1D CNN



```
Classification Report for sparse_categorical_crossentropy:
              precision    recall  f1-score   support

           0       0.00      0.00      0.00         3
           1       0.00      0.00      0.00         7
           2       0.88      1.00      0.93       134
           3       0.00      0.00      0.00         9

    accuracy                           0.88       153
   macro avg       0.22      0.25      0.23       153
weighted avg       0.77      0.88      0.82       153
```



Performance Overview:

- Precision, Recall, and F1-score: For both categorical_crossentropy and sparse_categorical_crossentropy, the precision, recall, and F1-score for class 0 and class 1 are all 0. This indicates that the model did not correctly classify any instances for these classes.
- For class 2, the precision, recall, and F1-score are relatively high, indicating that the model performed well in classifying instances belonging to this class.
- For class 3, similar to classes 0 and 1, the precision, recall, and F1-score are all 0, suggesting that the model failed to correctly classify any instances for this class as well.
- Accuracy: The overall accuracy for both categorical_crossentropy and sparse_categorical_crossentropy is 0.88, indicating that the model correctly classified 88% of the instances across all classes.
- Support: The support refers to the number of actual occurrences of each class in the dataset. For class 2, which has the highest support (134), the model's performance metrics are more reliable compared to classes with lower support (e.g., 0, 1, and 3).
- Macro Average and Weighted Average: The macro average calculates the metrics for each class independently and then takes the average, giving each class equal weight.
- The weighted average calculates the metrics for each class and then takes the weighted average based on the number of true instances for each class.

5. Discussion of Best results

In this chapter, we evaluate the experiments conducted to discover the most efficient NLP algorithms and sequence classification strategies for the given dataset. We thoroughly examine each component, from vectorization to hyperparameter optimization, to obtain the most favorable results.

5.1 Reflections on Optimal Vectorization Technique

Throughout the tests, the performance of TF-IDF and Count Vectorization approaches was rigorously examined when combined with a 1D CNN model. Despite the theoretical advantages that TF-IDF may have in weighting the relevance of tokens, practical research revealed no substantial advantage over Count Vectorization in terms of model performance. Both strategies failed to improve the model's prediction ability for minority classes. This convergence indicates that, for this specific dataset and task, the vectorization technique is less of a success factor than expected. Instead, fixing dataset imbalances or feature engineering may be more effective ways to improve model performance.

5.2 Insights into Successful NLP Algorithms

When comparing the 1D CNN and BiLSTM models, it was clear that the 1D CNN model performed better in recognising the 'B-O' class. However, neither approach performed well at detecting the less common 'B-AC', 'B-LF', and 'I-LF' classes. This discovery suggests that, while 1D CNNs are effective at capturing local patterns, more advanced models, like as transformer-based architectures, may be able to provide improved class classification by using contextual embeddings. This suggests a potential avenue for further investigation, implying that incorporating transformer-based models may improve performance in capturing intricate relationships within the data and enhancing the model's ability to distinguish between different classes, particularly those that are less frequent.

5.3 Assessment of Optimal Hyperparameter Strategy

Hyperparameter optimisation is critical for improving model performance. A comparison of Grid Search and Random Search approaches for hyperparameter tuning revealed that Grid Search performed marginally better in optimising the 1D CNN model. Despite this small gain, overall performance remained unsatisfactory. These findings highlight the significance of taking a more sophisticated strategy, maybe using approaches like Bayesian optimisation or genetic algorithms. These approaches have the ability to navigate the hyperparameter space more efficiently, resulting in more customised models and higher overall performance.

5.4 Comparative Analysis of Loss Functions

The analysis of different loss functions, particularly Categorical Cross-entropy versus Sparse Categorical Cross-entropy performed similarly to the 1D CNN model. This shows that the model's restrictions could be due to variables other than the loss function chosen, such as class imbalance or dataset characteristics. The results for optimizers did not show a clear superior selection, highlighting the possibility of future investigation into the learning dynamics. Incorporating unique loss functions or advanced optimizers may provide opportunities for improving model performance.

6. Conclusion

In this report, we conducted a thorough exploration of various natural language processing techniques aimed at addressing the complex task of sequence classification within scientific texts. Each experiment, spanning from vectorization strategies to hyperparameter optimization approaches, provided valuable insights into model behavior and performance.

While neural networks consistently demonstrated high accuracy in identifying standard text ('B-O' class), they struggled to effectively classify abbreviations and their long forms ('B- AC', 'B-LF', 'I-LF'). This highlights a significant gap in the models' capability, which appears to stem from issues related to data representation and distribution rather than inherent limitations in their learning abilities.

Our analysis of hyperparameter optimization techniques underscored the importance of a targeted search within the hyperparameter space. Although Grid Search showed slightly better performance compared to Random Search, the overall improvement was modest. This prompts consideration of alternative optimization methods in future research endeavors.

Regarding loss functions, the comparison between Categorical Cross-entropy and Sparse Categorical Cross-entropy revealed minimal differences in performance, suggesting that improvements in other areas of model tuning may yield more substantial gains. It's evident that addressing class imbalance and optimizing feature extraction methods are critical for further advancement.

In conclusion, this report lays the groundwork for future research endeavors. It recommends exploring advanced model architectures, customized loss functions, and enhanced data representation techniques. The integration of transformer-based models, attention mechanisms, and domain-specific embeddings holds promise for significant progress. Achieving a balanced precision and recall across all classes remains a challenging yet essential goal in natural language processing, requiring both computational rigor and innovative approaches to model development.