

# NATURAL LANGUAGE PROCESSING (COMM061)

## - Group 29

[1]Rizwan Bagdadi: 6899585	,[2] Ilakkiyavarsha Kannappan: 6900727
,[3] Lokesh Khadke: 6899997	[4]Mohan Sree Vijayakumar: 6903009

### **Abstract:**

This paper explores the task of biomedical token classification, specifically detecting abbreviations (AC) and their corresponding long forms (LF) in scientific texts, using the PLOD-CW-25 dataset. It explores models from classical machine learning approaches like Hidden Markov Models, Conditional Random Fields, and Support Vector Machines, progressing to deep learning models such as BiLSTM and BiLSTM + CNN, and finally explores modern transformer-based models including RoBERTa, BioBERT, and SciBERT. We also examined large language models in zero-shot and few-shot settings using prompt-based inference. Our findings show that while classical machine learning like CRF performed well with feature engineering, transformer models, like BioBERT, performed significantly better with an F1 score of ~0.84 due to domain-specific pretraining. We then deployed BioBERT in a prototype web application, demonstrating its effectiveness in real-world biomedical abbreviation and long-form detection tasks.

## Contents

<b>1. Introduction.....</b>	<b>3</b>
<b>2. Dataset Overview .....</b>	<b>3</b>
<b>2.1. Dataset Analysis and Evaluation .....</b>	<b>4</b>
<b>2.2. Data Preprocessing .....</b>	<b>5</b>
<b>3. Experimentation.....</b>	<b>7</b>
<b>3.1. Classical Machine Learning Approach.....</b>	<b>7</b>
<b>3.2. Deep Learning Models Approach.....</b>	<b>7</b>
<b>3.3. Transformer &amp; LLMs Approach .....</b>	<b>8</b>
<b>4. Results and Evaluation .....</b>	<b>9</b>
<b>4.1. Classical Machine Learning Results .....</b>	<b>9</b>
<b>4.2. Deep Learning Models Results .....</b>	<b>11</b>
<b>4.3. Transformer &amp; LLMs Results .....</b>	<b>12</b>
<b>5. Error Analysis .....</b>	<b>13</b>
<b>6. Deployment.....</b>	<b>14</b>
<b>7. Evaluation &amp; Conclusion .....</b>	<b>15</b>
<b>8. References.....</b>	<b>16</b>

## 1. Introduction

Abbreviations and long forms are often used in scientific and technical literature, especially in the field of biomedical research, where they are used to refer to complex terms in a simple form, making it easier and quicker for readers to read and understand the text. The task of detecting abbreviations involves identifying tokens that are short-form abbreviations (AC) and linking them to their corresponding long-form (LF) expansions. Automatically identifying these entities is extremely important in improving information retrieval, summarisation, and for automated understanding systems. The task of classifying/labelling each individual word (token) in a sentence with a specific class as “abbreviation” or “long-form” is a key part of the problem. Solving this problem is important as resolving these abbreviations can significantly improve the performance of Natural Language Processing (NLP) tasks, such as information retrieval and machine translation [1]. In the biomedical field, abbreviations are extremely common, such as BP for blood pressure and HIV for Human Immunodeficiency Virus, and this can introduce loss of information if they are not expanded.

This report offers a detailed examination of different NLP methods, utilising the PLOD-CW-25 dataset from `surrey-nlp` on Hugging Face, a subset of biomedical texts where each sentence has been annotated for abbreviations and long-form detection [2]. We investigate and compare multiple approaches to abbreviation and long-form detection, ranging from classical machine learning methods to modern deep learning models. We implemented three sequence labelling methods: a Hidden Markov Model (HMM), a Condition Random Field (CRF), and a Support Vector Machine (SVM) classifier. We then developed deep neural network models for the task, including a basic BiLSTM (Bi-directional LSTM) tagger and enhanced BiLSTM with CNN feature extraction. Finally, we experimented with transformer-based models like RoBERTa, BioBERT, SciBERT and LLM prompting. By comparing these approaches, we aim to understand how well these different techniques perform on limited biomedical data, and what their trade-offs are in terms of precision/recall.

## 2. Dataset Overview

The dataset used in this experiment, PLOD-cw-25, focuses on the scientific domain and includes 2,400 sequences; 2,000 for training, 150 validations, and 250 test. Each sequence is a sentence, or short paragraph, containing one or more abbreviations or long form. The text already contains tokens, where each token is annotated with a Part-of-Speech (POS) tag, and a named entity tag indicating “B-AC” for abbreviation tokens, “B-LF/I-LF” for long form tokens, or “O” for outside/non-entity tokens. The bio tagging scheme marks the beginning (B-) and inside (I-) of each entity. For example, in the sentence

“MRR , Mortality rate ratio ; TBI , traumatic brain injury” [2]

the tokens would be labelled as:

POS tags: “NOUN, PUNCT, NOUN, NOUN, NOUN, PUNCT, PROPN, PUNCT, ADJ, NOUN, NOUN, PUNCT” [2]

NER tags: “B-AC, O, B-LF, I-LF, I-LF, O, B-AC, O, B-LF, I-LF, I-LF, O” [2]

In this example, MRR is an abbreviation for Mortality Rate Ratio and TBI for Traumatic Brain Injury. The dataset aligns the abbreviations with their long form when they co-occur in the sentence. There are some cases where the abbreviations appear by themselves, but they are still labelled as AC or LF based on the context of the whole article.

2.1. Dataset Analysis and Evaluation

A deep analysis of the dataset was conducted in order to gain a deep understanding of the datasets characteristics and to inform modelling decisions. This involved examining the distribution of entity labels, sequence lengths, and POS tag frequencies.

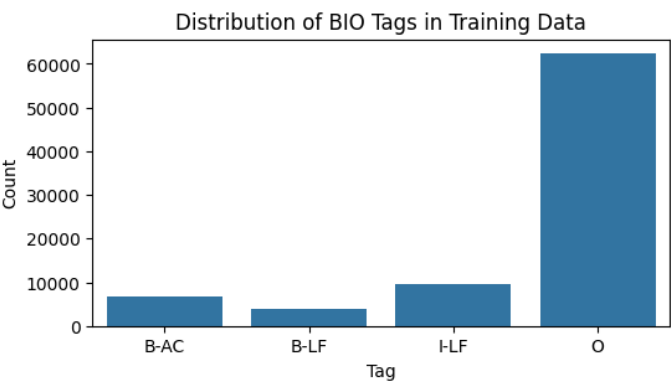


Figure 1. Token Label Distribution

As shown in Figure 1, most of the tokens are labelled as “O”, indicating that they do not belong to either abbreviations or long forms. In contrast, entity labels such as “B-AC”, “B-LF”, and “I-LF” appear far less frequently, showing that the dataset is highly imbalanced. This imbalance has several implications for model performance, as it can cause the models to overfit to the majority class, leading to accuracy scores that don’t reflect the model’s true ability to identify abbreviations and long forms. As relying on accuracy for a performance metric would be misleading, we decided to use evaluation metrics such as precision, recall, and F1-score to more accurately assess model performance. This prevents misleading interpretations that could possibly arise from using accuracy alone.

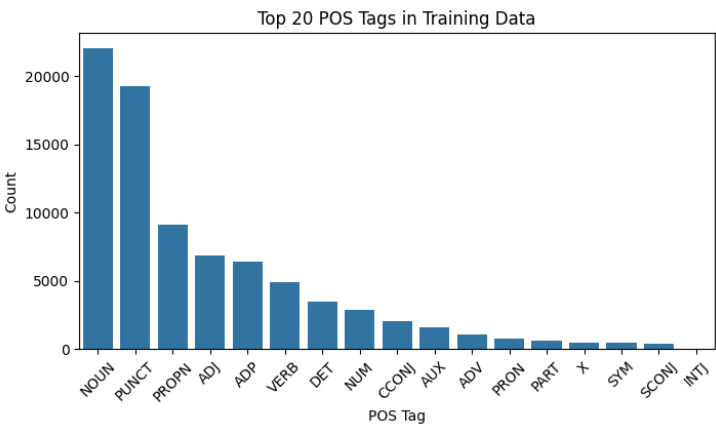
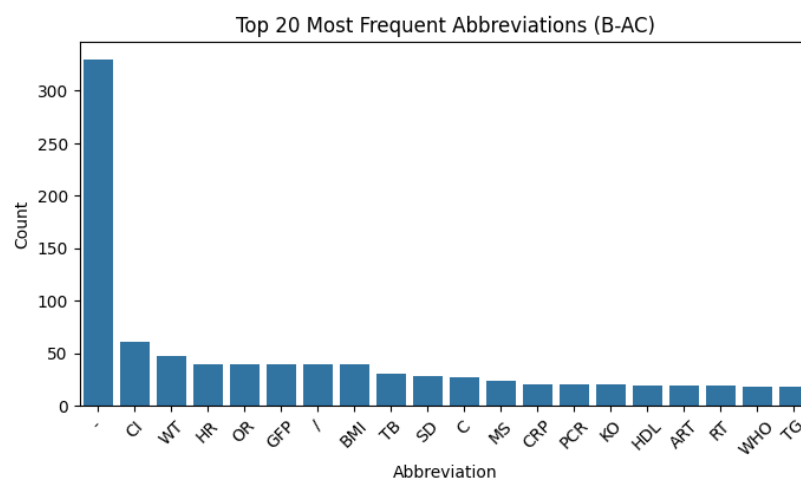


Figure 2. Top 20 POS tags in the training data

By analysing the POS tags of the labelled tokens, as show in Figure 2 illustrates the top 20 part-of-speech (POS) tags found in the training data, show that tags such as NOUN, PUNCT, and PROPEN appear much more frequently than others, which relates to the nature of the biomedical literature. Additionally, abbreviations are typically composed of capital letters, which the POS tagger often labels them as NOUN. On the other hand, long-form tokens can be mostly nouns or adjectives that form a descriptive phrase. For example, in “traumatic brain injury (TBI)”, the long-form tokens “traumatic” (ADJ), “brain” (NOUN), “injury” (NOUN) form a noun phrase.

POS tags are useful for the classical machine learning models, like CRFs and SVMs, as these models don’t learn language patterns by themselves, they rely on extra features, like POS tags, to understand the sentence structure.



**Figure 3.** Top 20 most frequent abbreviations

Figure 3 displays the top 20 most frequent abbreviations labelled with the B-AC tag in the dataset. The most common abbreviations are “CI” (Confidence Interval), followed by “WT” (Wild Type), and then “HR” (Hazard Ratio), which are common terms to be found in biomedical literature. Their high frequency in the dataset means that the models are more likely to learn to identify them much more easily. After the top 3, it appears that the frequency of other abbreviations remains fairly constant, meaning that the dataset has a reasonable variety of abbreviations.

To summarise, the dataset provides a valuable yet compact collection of different abbreviation and long-form instances. In the following sections, we outline the experimental approaches that we used to address this task, ranging from classical models that rely on feature engineering to more advanced neural architectures capable of learning contextual patterns directly from the data.

## 2.2. Data Preprocessing

Before we could use the dataset with our models, we had to ensure that the data would be compatible with each of the model approaches. Preprocessing was a key step in our token classification pipeline, as it allowed each model to work with the dataset. Although all the experiments shared the base

preprocessing of loading and structuring the dataset using Hugging Face's datasets library, each model required its own specific preprocessing steps to suit its architecture needs. One common step in all models is that we extracted the basic information such as the tokens, their POS tags, and their corresponding BIO-format labels, which were mapped to numeric values so that they would be easier to work with during training and evaluation stages. After this, the preprocessing was adapted based on the model used.

For the classical machine learning models, the preprocessing was heavily reliant on explicit feature engineering. In the HMM experiment, each word was converted into a numeric form using a hashing method, which helped keep the inputs compact, which is important as HMMs rely on matching sequences of symbols to predict tags. Whereas for the CRF model, each word was represented in a dictionary of its features, such as its lowercase form, suffixes, capital letters, whether it's a number, and its POS tag. We also included similar features from nearby words in order to give the model some more context. These steps helped the CRF model understand the structure of the sentence better. For the SVM model, we utilised a sliding window approach to create flat feature vectors for each token by integrating it with its surrounding context. These features derived from the sliding window were vectorised through a TF-IDF representation, allowing the model to perform token-level predictions using conventional classification methods.

For the deep learning models, each sentence was tokenised and lowercased, and stop words were removed to reduce noise. However, in the final version of the experiment, stop word removal was disabled to preserve meaningful biomedical terms and abbreviations. This step ensured contextually important tokens are retained for sequence learning. Also, for model compatibility, tokens were converted to integer indices using a vocabulary built from the training set. Labels were similarly mapped to numeric IDs. Both token and label sequences were padded to a fixed length of 100 to maintain uniform input shape. This structured formatting enabled batch processing and ensured seamless integration with embedding layers and sequence models during training. This processed dataset was then used to train BiLSTM and BiLSTM + CNN models.

For the transformer-based models, pre-processing took advantage of pretrained tokenisation schemes provided by the Hugging Face Transformers library. Specifically, we used model specific tokenisers, such as Byte-Pair Encoding (BPE) for RoBERTa and WordPiece for BioBERT and SciBERT. These tokenisers turn text into sub-word units that align with model inputs that maintain semantic space. Another key consideration during pre-processing was label alignment, as the sub-word tokenisation could cause splits, aligning the label of the original token to the first sub-word and pile the special ignore index (-100) on the other sub-words, so we do not mistakenly cause compute loss. This ensured correct label tracking while also preventing the model from mistakenly computing loss over irrelevant positions. Additionally, sequences were padded to a fixed length of 128 tokens and accompanied by attention masks, which indicated which parts of the input were actual tokens compared to padding. These preprocessing steps ensured the transformer models could be trained efficiently and evaluated consistently with different input lengths and structures.

Overall, these different preprocessing pipelines ensured that each experiment setup received their inputs in a form optimised for its architecture. By doing this, we were able to maximise the models'

ability to learn meaningful patterns from the data. This all played a crucial role in preserving the integrity of the dataset and in enabling fair, effective comparisons across the experimental setups.

### **3. Experimentation**

#### **3.1. Classical Machine Learning Approach**

For our first experiments, we implemented three classical sequence labelling approaches: a Hidden Markov Model (HMM), a Conditional Random Field (CRF), and an SVM-based classifier (SVM). These models were chosen to explore how well traditional machine learning methods could perform the abbreviation and long-form detection task using manual features and simpler architectures.

The HMM model approached the sequence labelling task as a generative issue. In this framework, each label (such as B-AC, B-LF, etc.) represents a hidden state, while each word serves as an observed symbol. The model determines the likelihood of transitioning between one tag to another, and the probability of each tag producing a specific word. The HMM then utilises the Viterbi algorithm to predict the most probable sequence of tags, evaluating the most likely paths through the sequence.

In contrast, the CRF model approached problem from a from a discriminative standpoint. Instead of focusing on the generation of sequences, it directly estimates the probability of a tag sequence in relation to a set of features. For each individual token, we developed an extensive set of features and used ‘L-BFGS’, which is a standard optimisation technique used to train the CRF. Then, to find the best hyperparameters, we used a randomised search, with cross-validation. After fine-tuning and training, the CRF was used to predict labels by taking advantage of the learned relationships between neighbouring tags and features.

For the final experiment, within the classical machine learning approach, we used an SVM model. The model approached token classification by making a sequence of independent predictions. We implemented a sliding window technique, which allowed us to extract features from the current token as well as its neighbouring words. These features were then converted into numerical vectors, using the TF-IDF method, which considers both the frequency of terms and their distinctiveness within the corpus. The resulting sparse feature vectors were then used to train a linear SVM classifier. Similar to the CRF, we did hyperparameter tuning using the randomised search to improve the classifier's performance.

By experiment with these three classical techniques, we were able to examine the effectiveness of classical machine learning methods and how they handle the complexities associated with biomedical token classification.

#### **3.2. Deep Learning Models Approach**

Next, we conducted experiments utilising deep learning sequence models, particularly Bi-directional LSTM networks, which are widely recognised for NER tasks. The primary benefit of deep learning models is their ability to extract abstract features from the data, such as word embeddings and contextual representations, without relying on manual feature engineering. However, with our dataset, we anticipated that the deep models might underperform classical methods like CRF unless properly tuned. All neural network models were developed using TensorFlow/Keras and trained on Google Colab with GPU support.

To assess the effectiveness of deep learning for token classification, we developed and tested three models: a baseline BiLSTM, an improved BiLSTM + CNN, and a Tuned BiLSTM. Each of the models shared a common architectural foundation but incorporated distinct variations to test specific hypotheses.

For the baseline BiLSTM model, each token was represented by a word embedding. We initialised the frozen embedding layer using 300-dimensional pre-trained GloVe vectors, which provide general word representations. A BiLSTM layer with 128 units was added to capture forward and backward context, followed by a dropout layer with a rate of 0.5 to prevent overfitting. A TimeDistributed dense layer with softmax activation produced token-level predictions. For tokens not found in GloVe (out-of-vocabulary), random vectors were used.

To enhance feature extraction, we implemented a hybrid BiLSTM + CNN model. Here, a convolutional layer with 64 filters and a kernel size of 3 was applied to the word embeddings before feeding them into the BiLSTM. This CNN layer helped capture local context and short-range dependencies more effectively. The remaining architecture remained the same as the baseline BiLSTM.

The Tuned BiLSTM model introduced key hyperparameter improvements. Dropout was reduced to 0.3, the learning rate was lowered to 0.0005 using the Adam optimiser, and the batch size was increased to 64 for improved training stability. Additionally, we applied EarlyStopping and ReduceLROnPlateau callbacks to stop training when validation loss stopped improving and to reduce the learning rate dynamically.

Each model was trained for a maximum of 10 epochs. The BiLSTM and BiLSTM + CNN used a batch size of 32, while the Tuned BiLSTM used 64. All models monitored validation loss during training to prevent overfitting and ensure better generalisation. During inference, the test data was preprocessed to match the training format, tokens were lowercased, mapped to indices, and padded to a fixed length of 100. The trained models then generated probability distributions over all possible labels for each token. The final predicted label for each token was selected using argmax and mapped back to its string label (B-AC, B-LF, I-LF, O). Performance was evaluated using precision, recall, and F1-score, computed with the seqeval library.

### **3.3. Transformer & LLMs Approach**

For our final set of experiment, we explored transformer-based models and large language models (LLMs) for token classifications. In this experiment, we included three transformer models: RoBERTa, BioBERT, and SciBERT, as well as zero-shot and few-shot prompting approaches using Gemma-7B-IT. Each experiment was designed to investigate how well different types of pretrained models adapt to the biomedical domain and token-level sequence labelling.

For the fine-tuned transformer models, we began with three transformer models: RoBERTa, BioBERT, and SciBERT, of which all are built upon the BERT architecture. These models use self-attention to capture long-range dependencies across tokens, allowing them to contextualise each word in relation to the entire sentence.

RoBERTa (Robustly Optimised BERT Pretraining) is a transformer model designed for general domains, trained on a large and diverse corpus. It improves upon BERT by getting rid of the Next Sentence Prediction (NSP) objective and using dynamic masking and larger batch sizes during



pretraining. The tokeniser for RoBERTa uses BPE which helps it effectively capture sub-word structures. BioBERT is also based on the BERT architecture, but it is further pretrained on biomedical text. This domain-specific pretraining allows it to better recognise biomedical terminology and abbreviations. SciBERT also follows the BERT architecture but is trained from scratch on scientific literature from various scientific fields. In contrast to BioBERT, which builds on BERT's vocabulary, SciBERT implements a unique tokenisation method to more effectively capture domain-specific language patterns.

These transformer fine-tuned models all started with initialising the models using 'AutoModelForTokenClassification', from the 'transformers' library, which added a classification head to the final hidden states of the transformer. The model was configured to predict one of four labels, being: O, B-AC, B-LF, or I-LF. We used the Hugging Face Trainer API in order to handle training, evaluation, and metric computation. Each model was trained for up to 10 epochs, using the AdamW optimiser with weight decay and different learning rates. We experimented with different hyperparameters: learning rates between  $1e-5$  to  $3e-5$ , batch sizes of 8 or 16, and weight decay of 0.01 or 0.1. We iterated over these to find the most effective configuration. The fine-tuning of the models works to minimise the cross-entropy loss between predicted and ground-truth token labels. After each epoch, we validate the model to track convergence and prevent overfitting, while at the same time monitoring and potentially stopping training altogether if there is no improvement in validation loss via early stopping.

To further experiment and to test generalisation without fine-tuning, we then implemented a zero-shot setup using gemma-7b-it, which is a large instruction-tuned LLM. The approach uses a prompt-based inference, where the model is given natural language instructions and a list of tokens, and is asked to return the correct NER labels (O, B-AC, B-LF, I-LF). The first step of the process is loading the model and tokenisers with appropriate padding and label mappings. For each input sequence, it is split into smaller chunks to fit the model's input size. The model then generates predictions, which are then processed to ensure they align with the token length.

We then implemented a few-shot learning setup using the Gemma-7B-IT language model, where we provided the model with a small number of labelled examples, which were taken from the training set to represent abbreviations and long forms. After giving the prompt examples, the model was given a new sequence of tokens and asked to generate a list of labels using the same structure. The generated predictions are then processed and evaluated using token-level classification metrics.

## 4. Results and Evaluation

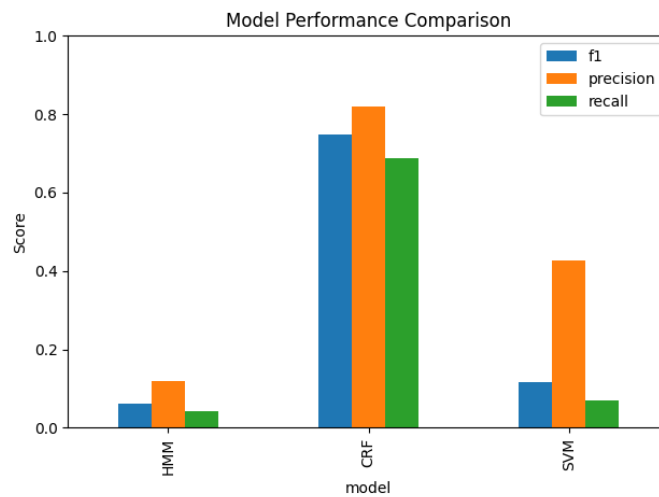
### 4.1. Classical Machine Learning Results

For the implementation, we utilised the seqeval library to assess these models on both the validation and test datasets [3]. Seqeval calculates precision, recall, and F1 scores specifically for sequence tagging tasks, ensuring that an entire long-form phrase is accurately tagged to be considered a true positive. We present the token-level precision and recall for the AC and LF classes, which corresponds to entity-level in this context, as each abbreviation or long-form entity encompasses one or more tokens.

Model	F1	Precision	Recall
HMM	0.058420	0.094788	0.042220
CRF	0.718902	0.786235	0.662236
SVM	0.108391	0.408236	0.064113

**Table 1.** Results from Classical Machine Learning Tests

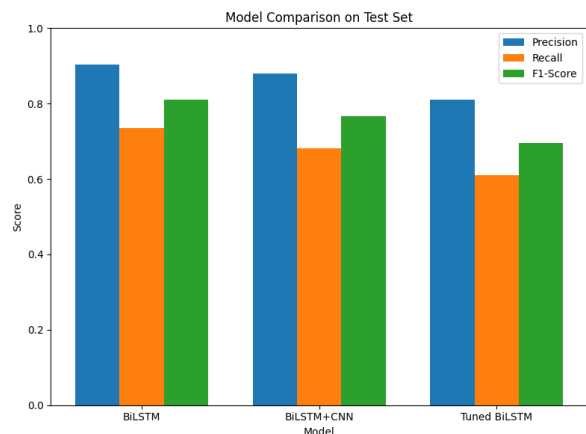
As expected, and as seen in Figure 1 and Table 1, the HMM had the poorest performance, with F1 below 0.1. It managed to only identify a small fraction of abbreviations (5% recall) and long forms, most likely due to its bias toward always predicting O, which lead to high precision on the few it did predict, but also low recall. The SVM had improved over HMM by using informative features. Its precision was moderate, with a score of 0.4, meaning half of the tokens it predicted as abbreviations were correct, but the other half were false alarms. The CRF was the best of the classical model, as it significantly improved recall compared to SVM, while maintaining a decent precision of ~79%, yielding an F1 of ~0.72.



**Figure 1.** Results from Classical Machine Learning Approach

Overall, the CRF performed best among the classical models. Its F1 score of ~0.71 was far higher than the SVM's ~0.11 and the HMM's ~0.06. By using a fixed random seed to ensure reproducibility, the training of the CRF demonstrated considerable stability following hyperparameter tuning. In contrast, the HMM frequently produced overly simple results and the SVM required careful tuning of context to avoid missing labels. In summary, the CRF's ability to use overlapping features and learn label dependencies gave it a clear advantage, and the HMM's simplicity proved to be a huge disadvantage, particularly in dealing with imbalanced labels, and the SVM's bag-of-words context had high precision but significantly low recall. These results show why the CRF emerged as the most effective classical model for our dataset.

## 4.2. Deep Learning Models Results



**Figure 2.** Bar Chart of Results from Deep Learning Model Tests

For the deep learning models, each model was trained using the preprocessed training set, with validation performance monitored after each epoch. The inclusion of callbacks such as EarlyStopping and ReduceLROnPlateau allowed the training to adapt dynamically, stopping when no improvement was observed and reducing the learning rate to encourage convergence. The training process was stable across all models, with validation accuracy consistently exceeding 94% by the final epochs. Precision, recall, and F1-score were calculated at the token level using the seqeval evaluation library. All three models demonstrated high precision, especially in identifying abbreviations (B-AC), while recall varied more significantly, particularly for long-form labels (B-LF, I-LF).

Model	Precision	Recall	F1-Score
BiLSTM	0.88	0.69	0.77
BiLSTM + CNN	0.88	0.70	0.78
Tuned BiLSTM	0.88	0.67	0.76

**Table 2.** Precision, Recall, and F1-Scores for Each Deep Learning Model

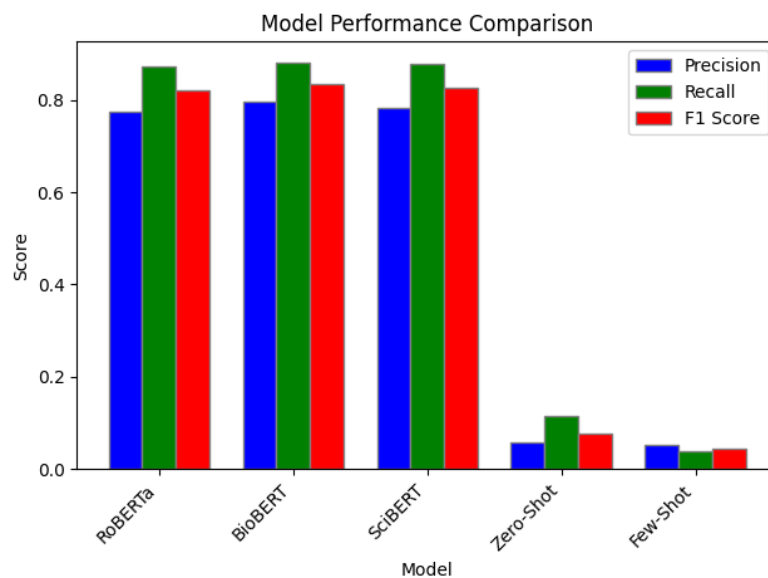
Based on the experimental findings, as seen in Table 2 and Figure 2, all three models achieved strong performance in abbreviation and long-form recognition, with BiLSTM + CNN emerging as the top performer in F1-score. Adding a convolutional layer before the BiLSTM helped the model learn short-term patterns more effectively, slightly improving recall compared to the plain BiLSTM. The Fine-tuned BiLSTM also performed well, benefiting from trainable embeddings and learning rate scheduling. While precision remained consistently high across all models (~0.88), recall showed more fluctuation, indicating challenges in identifying multi-token entity spans. Most misclassifications involved missing or mislabelling B- and I-LF tokens, as supported by the error analysis. The use of training callbacks notably improved model convergence and helped maintain generalisation performance. Overall, the BiLSTM + CNN model offered the best balance between stability, accuracy, and sequence-level consistency.

### 4.3. Transformer & LLMs Results

Model	Precision	Recall	F1-Score	Accuracy
RoBERTa	0.7756	0.8735	0.8206	0.8206
BioBERTa	0.7953	0.8821	0.8356	0.8356
SciBERT	0.7814	0.8771	0.8255	0.8255
Zero-Shot	0.0582	0.1142	0.0766	0.766
Few-Shot	0.0536	0.0377	0.0442	0.0442

**Table 3.** Results of Transformer Model Tests

Table 3 shows a summary of the performance of the models assessed, reporting precision, recall, F1-score, Accuracy, evaluation loss, run-time, and samples per second for the models. The metrics are based on performance on the test set of 150 examples and metrics are computed on tokens that are not ignored.



**Figure 3.** Bar Chart of Results of Transformer and LLM Models

BioBERT clearly performed better than all other models in every metric, with the highest F1-Score (83.56) and Accuracy (83.56). This is likely because BioBERT was pre-trained on biomedical literature that is very similar to the medical text data set. SciBERT had similar outcomes but did not perform as well because it was pre-trained on scientific content, and it included medical literature but not exclusively. RoBERTa performed slightly lower than BioBERT and SciBERT, but its broad-based pre-training ended up being a strong comparison. There was little performance from the Zero-Shot and Few-Shot approaches and these results were expected, as there was no task-specific fine-tuning and inference errors stemming from variables that are not defined into the evaluation pipeline. All the saved logs also document a step-by-step error trace and can identify which test chunks of were affected for easier debugging. There were some incongruences in computational efficiency as well, with RoBERTa having the fastest run-time (0.6341s) and highest throughput (236.548 samples/s), likely due to the fine-tuned architecture.

## 5. Error Analysis

To understand the strengths and weaknesses of each of our models, we reviewed prediction errors for all of our experiments. We aimed to identify common failure patterns, and determine why they occurred, and reflect on whether any of the parameters required adjustment based on these observations.

```
Found 302 boundary errors

Sample Boundary Errors:

Sequence 0, Position 51:
Context: the intracellular multiplication
True: ['O', 'B-LF', 'I-LF']
Pred: ['I-LF', 'I-LF', 'O']

Sequence 1, Position 4:
Context: Would contraceptive prevalence
True: ['O', 'B-LF', 'I-LF']
Pred: ['O', 'I-LF', 'O']

Sequence 3, Position 37:
Context: or exponential weighted
True: ['O', 'B-LF', 'I-LF']
Pred: ['I-LF', 'I-LF', 'O']
```

**Figure 4.** Error Analysis of HMM

As seen in Figure 4, the HMM model recorded 302 boundary errors. A common failure was mislabeling B-LF tokens as I-LF. These errors likely stem from HMM's Markovian assumption, which restricts the modeling of dependencies to neighboring tokens and ignores longer-range dependencies. CRF showed 51 boundary errors, much less than HMM, due to its more comprehensive feature representation. Even though CRF had fewer boundary errors, its reliance on local features without deeper context led to a few misclassifications. SVM had the least amount of boundary errors, with 43.

```
Sequence 0, Token 32:
Context : ['of', 'the', 'lcv', 'is', 'not']
True Label : B-AC
Pred Label : O

Sequence 0, Token 51:
Context : ['via', 'the', 'intracellular', 'multiplication', '(']
True Label : B-LF
Pred Label : O

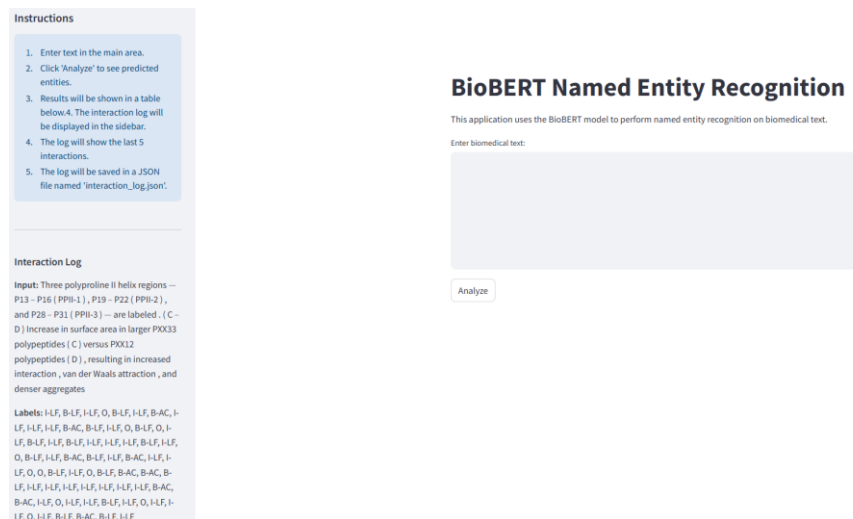
Sequence 0, Token 52:
Context : ['the', 'intracellular', 'multiplication', '(', 'icm)/defective']
True Label : I-LF
Pred Label : O
```

**Figure 5.** Error Analysis of BiLSTM

As seen in figure 5 the BiLSTM models kept frequently misclassified entity tokens like "lcv" and "intracellular multiplication" as non-entities (O). Most errors involved missing the start (B-\*) or continuation (I-LF) of long-form entities. This suggests difficulty in capturing span boundaries and context.

## 6. Deployment

For deployment, we chose BioBERT, a domain-specific transformer model pre-trained on large biomedical corpora, which produced the best performances at all three evaluation metrics and was therefore the best model for implementation. BioBERT's increased understanding of biomedical terminology allowed it to generalise well on new biomedical text, and it displayed the best performance for even the most complicated abbreviation contexts. BiLSTM performed similarly to BioBERT, but against all non-transformer architectures, BiLSTM remained the best model and closely followed BioBERT.



**Figure 6.** Screenshot of Interface

To provide an example of use, the BioBERT model was deployed in a report-only Streamlit web interface, as seen in figure 6, that allows users to input biomedical text and view predictions through interactive interface [4]. The model predictions (acronyms [AC] are shown in blue and their long form [LF] shown in orange) are coloured to provide real-time feedback in an understandable format as shown in figure 7. Use of colour improves the user experience and facilitates the verification of model outputs. This deployment represented the final phase in the experiment and demonstrates the application of sophisticated transformer-based models like BioBERT in biomedical NLP applications in the real world.

## BioBERT Named Entity Recognition

This application uses the BioBERT model to perform named entity recognition on biomedical text.

Enter biomedical text:

i love nlp

Analyze

Predicted Entities (All Labels):

	Token	Predicted Label
0	i	I-LF
1	love	B-LF
2	nlp	I-LF

Entity Visualization

i love nlp

Entity Visualization (Colorful)

I-LF love B-LF nlp I-LF

**Figure 7.** Screenshot of deployment example

Code for deployment: <https://github.com/mohansree14/Token-Classification>

Web app link : <https://token-classification.streamlit.app/>

## 7. Evaluation & Conclusion

Our experiments show a clear progression in token classification performance and show different levels of complexities from classical machine learning methods to modern transformers. In the first experiment, with classical models, the CRF model significantly outperformed the HMM and SVM models, confirming that CRF's ability to incorporate rich features yields better abbreviation detection. However, the CRF model was limited by its manual feature engineering and inability to capture long-range context. We then experimented with deep learning models, which significantly improved the results. The BiLSTM with a CNN character encoder achieved the highest accuracy within the deep learning models. This shows the importance of automatically learning contextual and sub-word features. The transformer base models provided the overall best performance. Especially BioBERT, with its domain specific pretraining, achieving the highest F1 score.

In terms of performance, the CRF model was fast to run on CPU, but its low accuracy limits its utility in practice. The BiLSTM + CNN model required more computation, with GPU training and careful hyperparameter tuning, but resulted in a higher F1 score of ~0.78, closely approaching the transformers' performance. In contrast, the transformer models, like BioBERT, required the most computational resources, but delivered the best results with an F1 score of ~0.84. We also saw that within transformers there were efficiency trade-offs, for example, RoBERTa had slightly faster inference throughput than BioBERT or SciBERT, which could have likely been due to optimisation and vocabulary differences. The large language models (LLaMA-2) had terrible results, both below 0.08, showing that the convenience of zero-shot predictions came at the expense of precision and reliability.

Implementing all these models led to many problems. With the deep learning models, one key issue was the initial removal of stop words, which inadvertently excluded valid entity tokens and negatively impacted performance, this was later corrected. Also, aligning labels with preprocessed tokens required careful handling to avoid misalignment during training, and tuning learning rates and batch sizes

introduced instability in early epochs, especially in the tuned BiLSTM model. Additionally, generating consistent evaluation outputs and visualising error cases demanded extra effort to automate and validate results across all models. With the transformer and LLM models, inference failures occurred for several test set chunks, mainly in the Zero-Shot and Few-Shot setups, as the evaluation script had undocumented variables (such as an expected sequence length) which resulted in incomplete predictions. The incorrect parsing of outputs, which involved errors that included invalid syntax and unsupported characters, meant that the results could not be compiled for a handful of examples, which left chunks needing manual inspection. There was a significant computational cost for transformer models, where prolonged evaluation time where the evaluation time took between 0.6341 seconds to 0.6571 seconds, and is indicative of the trade-off between computational and complexity efficiency.

Despite our challenges, our best models, BioBERT, shows promise for real-world deployment in abbreviation and long-form detection tasks. It demonstrated that it could fulfil the purpose of accurately identifying abbreviations (AC) and their expansions (LF) in biomedical text. With an F1-score of ~0.84, it captures most entities correctly, which we judge to be a strong outcome for this domain. This enabled us to use BioBERT in a prototype web application, where it generalised well to new biomedical sentences and reliably highlighted abbreviations and long-forms. With this level of accuracy, it could be useful for assisting a human reader, but in the case of a fully autonomous system, we would want an F1 score above 90%.

In Conclusion, our exploration of different models from HMMs to LLMs illustrates that illustrates that model architecture, domain adaptation, and resource availability all play key roles in having successful token classification. Despite our challenges in implementation and evaluation, our results show that with the appropriate data, model tuning, and evaluation strategies, high-performing NER systems for biomedical texts are feasible and effective. This reflects on the broader trend in NLP, where models are becoming more complex and computationally demanding, they are also become more powerful, resulting in new opportunities for automation in scientific and medical contexts.

## 8. References

- [1] Zilio, L., Saadany, H., Sharma, P., Kanojia, D. and Orâsan, C. (2022). PLOD: An Abbreviation Detection Dataset for Scientific Documents. [online] pp.20–25. Available at: <https://aclanthology.org/2022.lrec-1.71.pdf> [Accessed 16 May 2025].
- [2] Surrey (2025). PLOD-CW-25. [online] Huggingface.co. Available at: <https://huggingface.co/datasets/surrey-nlp/PLOD-CW-25>.
- [3] GitHub. (2023). sequeval. [online] Available at: <https://github.com/chakki-works/sequeval>.
- [4] Streamlit.app. (2025). Available at: <https://token-classification.streamlit.app/> [Accessed 16 May 2025].
- [5] J. Lafferty, A. McCallum, and F. C. N. Pereira, "Conditional random fields: Probabilistic models for segmenting and labeling sequence data," in *Proc. 18th Int. Conf. Machine Learning (ICML)*, Williamstown, MA, USA, 2001, pp. 282–289.
- [6] G. Lample, M. Ballesteros, S. Subramanian, K. Kawakami, and C. Dyer, "Neural architectures for named entity recognition," in *Proc. Conf. of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT)*, San Diego, CA, USA, 2016, pp. 260–270.
- [7] J. Lee, W. Yoon, S. Kim, D. Kim, S. Kim, C. H. So, and J. Kang, "BioBERT: A pre-trained biomedical language representation model for biomedical text mining," *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, Apr. 2020.



## Declaration

I confirm that the submitted work is my own work. No element has been previously submitted for assessment, or where it has, it has been correctly referenced. I have clearly identified and fully acknowledged all material that is entitled to be attributed to others (whether published or unpublished) using the referencing system set out in the programme handbook.

I agree that the University may submit my work to means of checking this, such as the plagiarism detection service Turnitin® UK and the Turnitin® Authorship Investigate service. I confirm that I understand that assessed work that has been shown to have been plagiarised will be penalised.

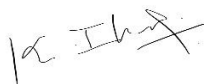
If in completing this work I have been assisted with its presentation by another person, I will state their name and contact details of the assistant in the 'Comments' text box below. In addition, if requested, I agree to submit the draft material that was completed solely by me prior to its presentational improvement.



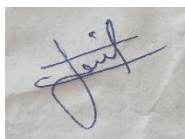
[1] Rizwan Bagdadi



[4] Mohansree Vijayakumar



[2] Ilakkiyavarsha Kannappan



[3] Lokesh Khadke