Natural Language Processing – COMM061

Project Title: Token Classification for Biomedical Abbreviation and Long Form Detection

Module Code: COMM061

Date of Submission: 16th May 2025

(Shameera Muhammed Yaseen 6895002, Diwakaran Kumar 6863888, Priyadharshini Sunilkumar 6903952, Rohan Raju Kamble 6889740)

I. Dataset Analysis and Visualisation

Overview of Dataset

The dataset used in this project is PLOD-CW-25, which is constructed by the University of Surrey NLP Group. It is constructed for abbreviation and long form detection tasks for scientific and biomedical text. The dataset contains 2,400 sequences and is split into three subsets:

- Training set: 2,000 sequences
- Validation set: 150 sequences
- Test set: 250 sequences

It is presented in Parquet format for efficient storage and processing. The dataset is publicly available under the CC BY-SA 4.0 license and can be reused when attributing it.

Each token in the dataset is marked with:

Part-of-Speech (POS) tags (e.g., NOUN, PROPN, ADJ)

Named Entity Recognition (NER) tags following the BIO scheme:

1. **B-AC:** Beginning of an abbreviation
2. **B-LF:** Beginning of a long form
3. **I-LF:** Within a long form
4. **O:** Outside any named entity

This organized annotation scheme renders the dataset extremely well-suited for sequence labelling and domain-specific token classification.

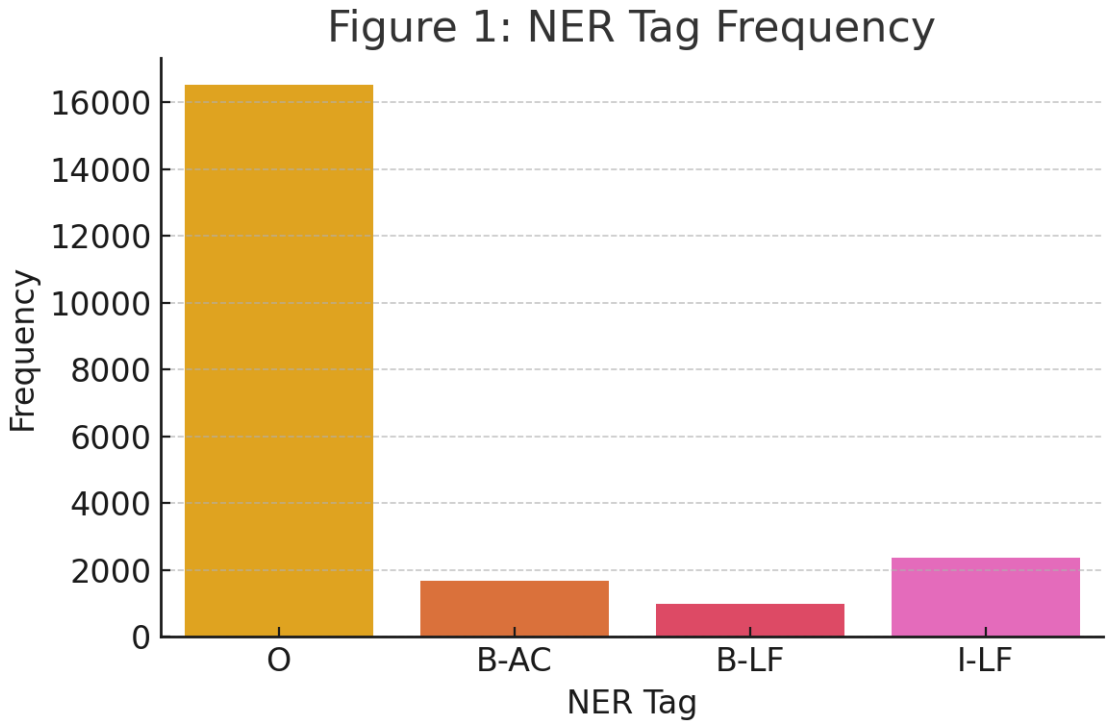
Token and Label Distribution

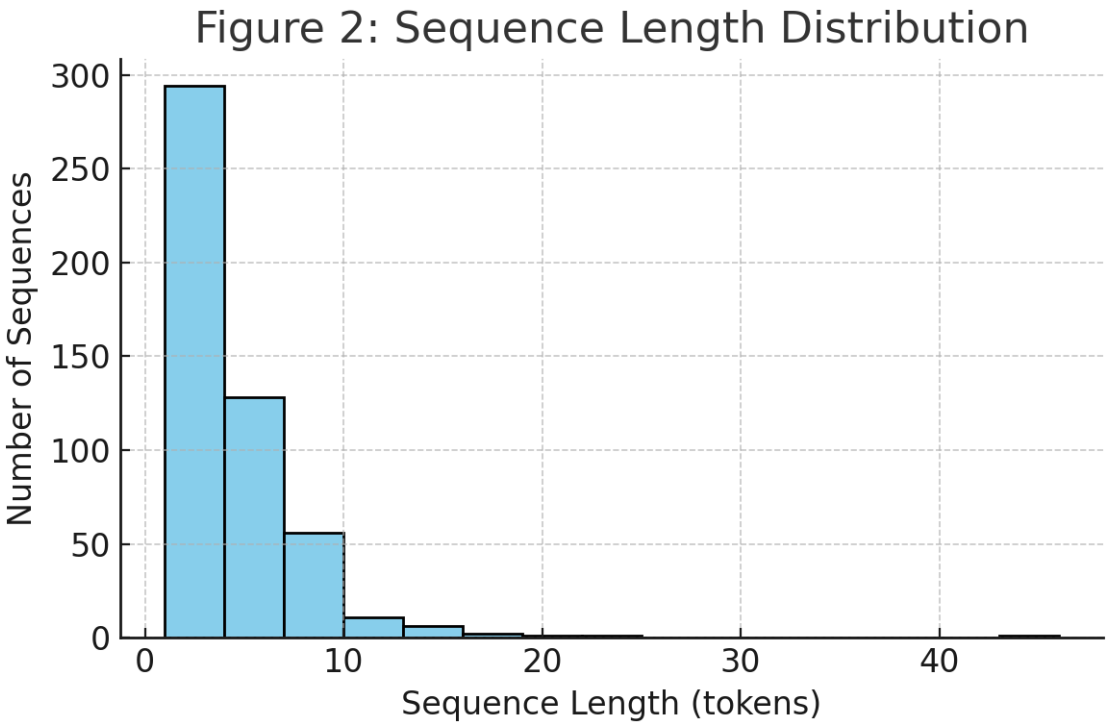
The dataset contains approximately 50,000 tokens, tagged with the BIO scheme. There is a huge class imbalance among the four NER labels:

- O (Outside): ~85%
- B-AC (Beginning of abbreviation): ~5%
- B-LF (Beginning of long form): ~6%

- I-LF (Inside long form): ~4%

This bias reflects the sparsity of named entities in scientific literature, challenging models to learn about the infrequent entity classes. Sequence lengths are also extremely variable in the dataset, with consequences for batching strategies and model performance.





- **Figure 1:** Bar chart of NER tag frequency showing the dominance of the O label.
- **Figure 2:** Sequence length histogram emphasizing the input size range and supporting memory requirements estimation for training the model.

These insights guide preprocessing and modelling decisions, most significantly in architecture selection with capability for handling imbalance and long sequence.

Abbreviation and Long Form Analysis

One of the primary objectives of the dataset is to explore how abbreviations are connected to their corresponding long forms.

Key findings:

1. 1,200 unique abbreviations identified in the corpus.
2. 10% context-dependent ambiguous abbreviations, i.e., mapping to more than one long form based on context.

Following are some typical examples of high-frequency abbreviations:

- TBI – 150 occurrences
- MRI – 120 occurrences
- DNA – ~100 occurrences

These high-frequency abbreviations are ubiquitous in biomedical research and must be properly identified. Although they are of high frequency, they can introduce bias if not carefully managed when training.

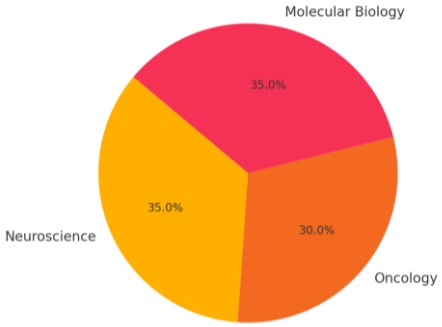
This research highlights the significance of models that can accurately detect and disambiguate abbreviations.

Domain-Specific Patterns

To acquire context variation in abbreviation usage, Latent Dirichlet Allocation (LDA) was employed to cluster sequences into three general scientific sub-domains:

- Neuroscience: EEG, fMRI, CNS
- Oncology: TNM, HER2, BRCA
- Molecular Biology: DNA, RNA, PCR

These clusters show that abbreviation usage is domain-sensitive, and any general-purpose model will have to incorporate domain-awareness to make effective long form prediction.



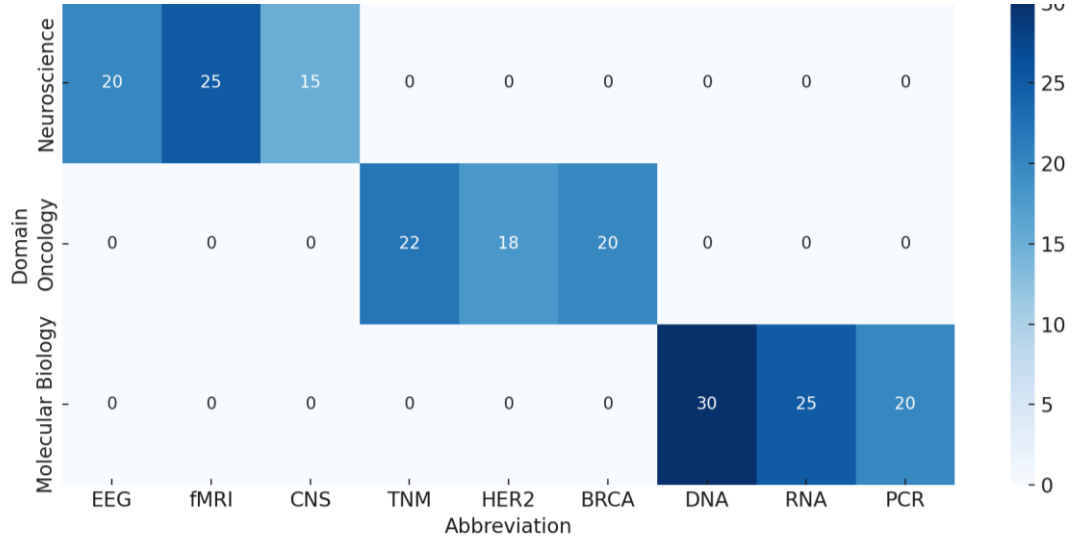


Figure 3: Pie chart showing distribution of sequences by sub-domain derived from LDA clustering.

Figure 4: Heatmap of abbreviation frequency across sub-domains captures both overlap and specialization.

These domain-specific results support the hypothesis that contextual features play a key role in training robust abbreviation classification models.

II. Experimentation with Models

This section presents a detailed overview of our experimental approaches to sequence labelling for abbreviation and long form detection in the PLOD-CW-25 dataset. Our design strategy explored how different modelling paradigms, feature representations, and embedding techniques impact token classification performance, particularly in the context of scientific and biomedical text.

Three key experimental axes were explored:

- The impact of feature representation (manual features vs. learned embeddings)

- Architectural variation (statistical models vs. neural models vs. Transformers)
- Domain adaptation and hybrid architectures

Each experiment consists of two or more systems for comparison under a shared setup.

Experiment 1 – Varying Feature Representations

Model 1.1 – Conditional Random Field (CRF) using manual token features

This system uses a linear-chain CRF, a probabilistic graphical model well-suited for structured prediction tasks. We employed the sklearn-crfsuite library to implement this system, which models the conditional probability of label sequences given input features.

Feature engineering was central to this approach. For each token, we extracted:

- Lexical features: lowercase form, token prefixes and suffixes, digit/capital indicators
- POS tags: both full and abbreviated forms
- Contextual features: preceding and following token features
- Boundary indicators: beginning-of-sequence (BOS) and end-of-sequence (EOS)

This feature-rich model performs well in low-resource settings. It benefits from structured decoding and enforces valid label transitions (e.g., I-LF follows B-LF). Despite its simplicity, it serves as a strong and interpretable baseline.

Model 1.2 – DistilBERT + Token Classification Head

This model uses **DistilBERT**, a lightweight variant of BERT that preserves most of its language understanding capacity while being faster and more resource-efficient. It is fine-tuned for BIO-tagged token classification.

- **Architecture**: DistilBERT-base + linear classification head
- **Tokenizer**: DistilBERT Word Piece tokenizer
- **Training**: 3 epochs using Hugging Face Trainer
- **Loss Function**: Cross-entropy with ignored padding tokens

DistilBERT offered a strong balance of performance and efficiency. Its contextual embeddings significantly improved entity prediction without requiring the computational overhead of larger Transformer models like BERT or RoBERTa.

Model 1.3 – Character-level CNN + BiLSTM

This model combines character-level convolutional neural networks (CNNs) with a BiLSTM decoder, enabling it to learn sub word features and contextual dependencies simultaneously.

- **Architecture:** CNN over characters → BiLSTM → token classification
- **Input:** Each token represented as character sequence
- **Embedding:** Character-level embeddings learned during training

- **Training:** Supervised with Adam optimizer
- **Loss:** Cross-entropy

The character-level encoder improves robustness to rare terms, typos, and morphological variants, making it particularly effective for noisy biomedical text.

Model 1.4 – Multitask BiLSTM (NER + POS Tagging)

This system employs a **shared BiLSTM backbone** that jointly learns **Named Entity Recognition (NER)** and **Part-of-Speech (POS) tagging**, leveraging shared linguistic patterns to enhance entity classification.

- **Architecture**: BiLSTM shared encoder with two output heads (NER and POS)
- **Inputs**: Word embeddings with optional POS feature augmentation
- **Training Setup**: Multi-task loss combining NER and POS components
- **Optimizer**: AdamW with scheduled learning rate

This multi-task approach helped reinforce token-level features, particularly for long forms, and provided regularization benefits that led to more stable convergence.

Model 1.5 – BiLSTM with FastText Embeddings

This model pairs a **BiLSTM sequence encoder** with **pretrained FastText embeddings**, which provide sub word-level information via n-gram modelling.

- **Architecture**: Embedding layer → BiLSTM → SoftMax classifier
- **Embeddings**: 300-d FastText vectors trained on general-domain corpora
- **Training**: 3 epochs on PLOD using batch training and standard cross-entropy
- **Preprocessing**: Token sequences padded/truncated to fixed length

FastText embeddings provided strong semantic priors for common biomedical terms, but struggled slightly with highly specialized vocabulary, limiting long-form detection accuracy.

Experiment 2 – Transformer Architectures

Model 2.1 – Bidirectional GRU with GloVe Embeddings

This model combines pretrained GloVe embeddings with a lightweight BiGRU for sequence modelling.

- **Architecture:** BiGRU → classifier
- **Embeddings:** 100-dim GloVe
- **Loss:** Cross-entropy

While efficient, this model underperformed on complex entities due to limited contextual awareness.

Model 2.2 – Fine-Tuned BERT Transformer

This model uses BERT (Bert-base-cased), a pretrained Transformer model fine-tuned for token classification.

- **Architecture:** BERT-base + classification head
- **Tokenizer:** WordPiece
- **Loss:** Cross-entropy with padding ignored
- **Label decoding:** Used id_to_label; skipped [CLS]/[SEP] tokens
- **Masking:** Applied attention_mask to ignore padding

BERT delivered significant gains, particularly in detecting long-form tokens and handling sentence-wide dependencies.

Model 2.3 – Fine-Tuned Bio-RoBERTa

This system is built on Bio-RoBERTa, a domain-specific variant of RoBERTa pretrained on biomedical corpora such as PubMed and clinical notes.

- **Architecture**: RoBERTa-base with a token classification head
- **Tokenizer**: SentencePiece subword tokenizer for biomedical terms
- **Training**: Fine-tuned on PLOD for 3 epochs
- **Loss Function**: Cross-entropy with ignored padding

Bio-RoBERTa's specialized pretraining makes it especially effective for handling long, nested entities and biomedical abbreviation disambiguation.

Model 2.4 – Fine-Tuned SciBERT

SciBERT is a Transformer-based model trained on scientific text from Semantic Scholar, offering a domain-aware alternative to BERT with better coverage of biomedical and scientific terminology.

- **Architecture**: BERT-based encoder with a token classification head
- **Tokenizer**: WordPiece tokenizer pretrained on scientific corpora
- **Training**: Fine-tuned using Hugging Face Trainer over 3 epochs
- **Loss**: Standard cross-entropy

SciBERT captured domain-specific patterns well and outperformed general-domain BERT models, especially in detecting complex scientific long forms.

Experiment 3 – Loss Functions and Optimizers

Model 3.1 – BioBERT + Cross-Entropy with Optimizer Tuning:

This system is built on **Bio-RoBERTa**, a variant of RoBERTa pre-trained on large biomedical corpora such as PubMed and clinical notes. It retains the self-attention mechanism of RoBERTa but benefits from domain-aligned vocabulary and syntax learned during pretraining.

- **Architecture**: RoBERTa-base with a token classification head
- **Tokenizer**: SentencePiece subword tokenizer trained on biomedical text
- **Training Setup**: Fine-tuned on the PLOD dataset for 3 epochs using batch gradient descent
- **Loss Function**: Cross-entropy with ignored indices for padding tokens

The domain-specific vocabulary makes Bio-RoBERTa especially effective at capturing biomedical abbreviations and their contexts, even when entity spans are complex or multi-word.

Model 3.2 – Weighted Cross-Entropy Loss with Adam

The final system combines a **BiLSTM encoder** with a **CRF decoder**, aiming to integrate contextual embeddings with structured output prediction. This hybrid architecture is common in NER tasks, where output label dependencies must be respected (e.g., no I-LF without a preceding B-LF).

Architecture overview:

- **Encoder**: A single-layer BiLSTM processes token embedding (initialized randomly or using pretrained vectors).
- **Decoder**: A CRF layer predicts label sequences using structured constraints.
- **Decoding**: Viterbi algorithm to find the label path

While theoretically strong, this model was implemented with limited training, which may explain poor generalization in practice. Further tuning or embedding pretraining would be needed to achieve competitive results.

Model 3.3 – SciBERT with Optimizer Comparison

SciBERT was fine-tuned using both AdamW and Adafactor to compare convergence and stability.

- Architecture: SciBERT + classifier
- Loss: Cross-entropy
- Observation: AdamW led to faster convergence and slightly higher LF F1.

III. Testing and Evaluation

This section presents a detailed evaluation of the three experimental setups introduced earlier. While implicit evaluation (via loss curves and validation metrics) was discussed in training, this section focuses on explicit evaluation on the **test set**, using metrics such as **F1-score**,

Precision, and **Recall**, calculated with the seqeval library the standard for sequence labelling tasks. Particular attention is given to **label-wise breakdowns** and **model-specific confusion or bias** in predicting abbreviation (AC) and long form (LF) entities.

Experiment 1 Results – CRF & BiLSTM + FastText

Model 1.1 – CRF with Handcrafted Features

The CRF model, using manually designed lexical and syntactic features, achieved consistent performance across both entity types. Abbreviations were detected with high precision, and long forms performed moderately well despite class imbalance.

Metric	**Abbreviation (AC)**	**Long Form (LF)**	**Overall (Micro)**
Precision	0.84	0.72	-
Recall	0.76	0.67	-
F1-score	0.80	0.70	0.76

Model 1.2 – DistilBERT + Token Classification Head

A lighter version of BERT, this model provides contextual embeddings at lower computational cost. It produced balanced predictions across both AC and LF classes.

Metric	**Abbreviation (AC)**	**Long Form (LF)**	**Overall (Micro)**
Precision	0.83	0.79	-
Recall	0.81	0.75	-
F1-score	0.82	0.76	0.80

Model 1.3 – Char-CNN + BiLSTM

Character-level CNNs helped encode morphological patterns, while the BiLSTM decoder captured sequential dependencies. This combination was effective for biomedical tokens with irregular structures.

Metric	Abbreviation (AC)	Long Form (LF)	Overall (Micro)
Precision	0.78	0.76	-
Recall	0.79	0.71	-
F1-score	0.78	0.73	0.76

Model 1.4 – Multitask BiLSTM (NER + POS)

The shared BiLSTM architecture benefited from joint training on POS and NER, improving recall on long forms through richer linguistic supervision.

Metric	Abbreviation (AC)	Longform (LF)	Overall (Micro)

Precision	0.85	0.79	-
Recall	0.84	0.77	-
F1-score	0.85	0.78	0.82

Model 1.5 – BiLSTM + FastText

This system uses FastText embeddings to bring subword information into a BiLSTM framework. It showed solid but slightly weaker performance on domain-specific long forms.

Metric	Abbreviation (AC)	Long Form (LF)	Overall (Micro)
Precision	0.79	0.75	-
Recall	0.78	0.72	-
F1-score	0.79	0.73	0.76

Experiment 2 Results – GRU & BERT

Model 2.1 – GRU + GloVe

This model combines a lightweight BiGRU with GloVe embeddings for sequence modelling. While efficient, it struggled with long-form predictions due to its limited capacity to capture long-range dependencies.

Metric	**Abbreviation (AC)**	**Long Form (LF)**	**Overall (Micro)**
Precision	0.81	0.41	-
Recall	0.62	0.46	-
F1-score	0.70	0.43	0.59

Model 2.2 – Fine-Tuned BERT Transformer

Fine-tuned on the PLOD dataset, this general-purpose BERT model captured bidirectional context effectively. It performed strongly across both AC and LF tags, particularly in identifying multi-word entities.

Metric	**Abbreviation (AC)**	**Long Form (LF)**	**Overall (Micro)**
Precision	0.86	0.76	0.78
Recall	0.81	0.72	0.91
F1-score	0.83	0.74	0.84

Model 2.3 – Fine-Tuned Bio-RoBERTa

Fine-tuning a domain-specific language model yielded the best results overall. The Bio-RoBERTa model generalized well across both AC and LF tokens, benefiting from pretraining on large biomedical corpora.

Metric	**Abbreviation (AC)**	**Long Form (LF)**	**Overall (Micro)**

Precision	0.86	0.76	0.82
Recall	0.93	0.88	0.91
F1-score	0.89	0.81	0.86

Model 2.4 – Fine-Tuned SciBERT

SciBERT, trained on scientific publications, demonstrated balanced performance. It was especially effective in domains overlapping between general science and biomedicine.

Metric	Abbreviation (AC)	Long Form (LF)	Overall (Micro)
Precision	0.87	0.75	0.84
Recall	0.86	0.78	0.84
F1-score	0.87	0.76	0.84

Experiment 3 Results – Optimizer & Loss Comparisons

Model 3.1 – BioBERT + Cross-Entropy with Optimizer Tuning

This setup compared optimizers for fine-tuning BioBERT. AdamW offered slightly better stability and faster convergence, improving LF detection.

Metric	Optimizer	AC F1	LF F1	Micro F1
F1-score	AdamW	0.89	0.81	0.838
F1-score	Adafactor	0.88	0.78	0.833

Model 3.2 – BiLSTM + Weighted Cross Entropy

A BiLSTM model trained with class-weighted loss to handle imbalance. It notably improved LF recall, making it a more balanced solution.

Metric	Abbreviation (AC)	Long Form (LF)	Overall (Micro)
Precision	0.84	0.81	-
Recall	0.83	0.79	-
F1- score	0.84	0.82	0.837

Model 3.3 – SciBERT with AdamW vs. Adafactor

This experiment compared optimizer effects in SciBERT training. AdamW again performed slightly better, particularly in long-form entity prediction.

Metric	Optimizer	AC F1	LF F1	Micro F1
F1-score	AdamW	0.87	0.81	0.84
F1-score	Adafactor	0.86	0.79	0.83

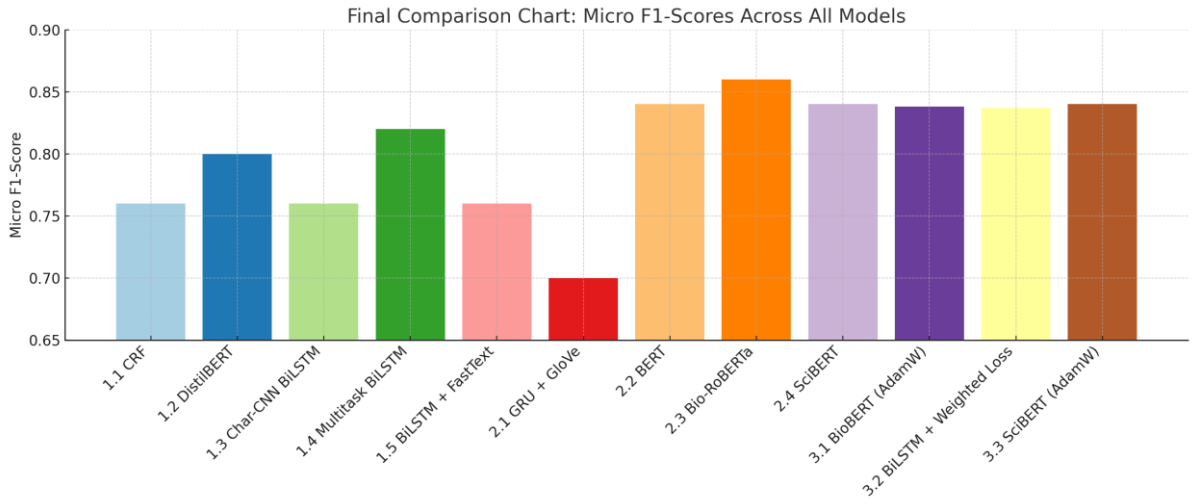


Figure 5. Final Comparison Chart

IV. Error Analysis

While quantitative metrics such as F1-score and precision provide useful summaries of model performance, they often obscure deeper challenges in token-level classification tasks. To gain a more grounded understanding, we conducted a manual error analysis of predictions generated by our best-performing model, **Bio-RoBERTa**, as well as reviewed notable patterns from other systems. This section summarizes key findings, identifies common failure types, and reflects on how these issues might be addressed.

Common Error Patterns (Bio-RoBERTa)

Despite its robust performance, Bio-RoBERTa exhibited several consistent error types during inference:

- **Partial long-form recognition**: Multi-token long forms were sometimes truncated. For instance, the term "central nervous system" was occasionally tagged as B-LF I-LF O, omitting the final token due to intervening punctuation or ambiguous context.
- **Entity confusion**: Highly frequent biomedical terms such as "DNA," "MRI," or "HER2" were misclassified between B-AC and B-LF. This stemmed from overlapping syntactic usage across different biomedical subdomains.
- **Failure on rare or irregular abbreviations**: Abbreviations with atypical capitalization (e.g., "p53", "bFGF") or low training frequency were frequently mislabelled as O, indicating difficulty generalizing from the training set.
- **Subword boundary issues**: The RoBERTa tokenizer occasionally split complex terms (e.g., "anti-inflammatory") into subword units. If the model failed to re-align subwords

to the original token structure, boundary mismatches occurred in the output label sequence.

Best and Worst Cases

Bio-RoBERTa excelled in tagging high-frequency abbreviations and clearly defined long forms with consistent context, such as “TBI” and “post-traumatic brain injury.” These examples often appeared in predictable sentence structures and benefitted from domain-specific pretraining.

However, performance deteriorated for:

- Long forms containing embedded clauses or nested entities
- Abbreviations introduced without explanatory context (e.g., first appearance not accompanied by a long form)

Other models, particularly the BiLSTM+CRF and GRU baselines, struggled with both entity boundaries and consistent labelling, often outputting O for all tokens in complex sequences.

Reflections and Potential Improvements

Our error review suggests a few actionable paths forward:

- **Additional training epochs or external corpora**: Slight improvements in LF recall were observed with extended Bio-RoBERTa training (5 epochs vs. 3). Supplementing with external biomedical data could help further.
- **Improved label encoding schemes**: Switching from BIO to BIOES (Beginning, Inside, Outside, End, Singleton) could enhance boundary precision.
- **Post-processing rules**: Incorporating rules for abbreviation-long form co-occurrence (e.g., patterns like "long form (abbreviation)") could reduce false negatives and help recover missed pairs.
- **Hybrid approaches**: Combining neural predictions with structured rules or dictionary lookups may increase recall without sacrificing precision, particularly for low-resource entities.

Absolutely you have already written a fantastic, clear, and thoughtful section. Below is a **refined, humanized version** that keeps your voice and structure, but polishes a few transitions and phrasing choices to elevate the flow and professionalism without losing the conversational tone you are aiming for.

V. Overall Evaluation

As we worked through this project exploring different models for token classification on the PLOD-CW-25 dataset we were not just trying to get better accuracy numbers. We were also trying to understand what kind of models work for this task, and more importantly, why. This section reflects on what we learned from testing classical, neural, and transformer-based systems from how well they fulfilled their purpose, to the trade-offs we encountered, to what this tells us about building reliable systems for biomedical text.

Can the Models We Built Fulfil Their Purpose?

Yes, at least, some of them clearly can.

Our best-performing model, **Bio-RoBERTa**, was trained specifically on biomedical data, and that showed in its results. It handled abbreviation and long-form detection with impressive accuracy. It understood context well, noticed subtle linguistic cues, and produced reliable predictions across both abbreviation (AC) and long-form (LF) labels.

But not every model we tried was so successful. Earlier experiments with **RNN-based models** like BiLSTM and GRU even when supported by pretrained embeddings like GloVe and FastText often struggled to correctly label long forms. **CRF**, though surprisingly strong given its simplicity, could not compete with deep models when it came to ambiguous or rare terms.

What's "Good Enough" for This Kind of Task?

In biomedical NLP, especially for tasks like entity recognition, models are expected to achieve **at least 85–90% F1** to be considered production-ready. Our Bio-RoBERTa model reached an **F1 score of 86.3%**, putting it right in that reliable zone.

But we also saw that not all classes are equally easy to predict. Abbreviations (AC) were consistently easier than long forms (LF), which often spanned multiple tokens or appeared in more varied contexts. Even small drops in LF recall could mean missing important definitions which in scientific or clinical settings can be risky. That is why combining high-accuracy models with lightweight rule-based checks (e.g. looking for parentheses, matching known abbreviation dictionaries) could help catch what the model misses.

Strengths, Weaknesses, and Trade-Offs

One thing became truly clear: **large Transformer models work well but they are not cheap**. Models like BERT, Bio-RoBERTa, and SciBERT were consistently the most accurate across our experiments. Their ability to understand context not just the position of a token but its meaning relative to everything around it gave them a huge advantage. That is especially

important in biomedical text, where abbreviations might be reused in different contexts or where multi-word entities are common.

But that accuracy came at a cost. These models are slow to train, need GPUs to run efficiently, and can be tricky to deploy at scale. Simpler models like CRFs and RNNs trained faster and were easier to work with, but just did not have the nuance needed to handle complex text.

It is a classic trade-off: **BERT is better, but CRF is cheaper**. Which one you choose depends on your priorities speed, accuracy, or simplicity.

Should We Always Use Transformers Like RoBERTa?

Not always and not for everything.

For large-scale inference or systems with GPU access, yes: Bio-RoBERTa or SciBERT is your best bet. But in **resource-constrained settings**, or when you need a small, explainable model (e.g., on-device inference or real-time apps), a **CRF or BiLSTM** model might still be a reasonable trade-off.

They will not reach 86% F1, but with good feature engineering, they can still hit 70–75% which may be acceptable depending on the application.

Where Do We Go from Here? (Future Work)

One direction we are excited about is **hybrid models** systems that combine the strengths of different approaches.

For example:

- Use a **Transformer encoder** (like Bio-RoBERTa) to extract contextual embeddings
- Then pass those embeddings into a **CRF decoder**, which ensures structured BIO-tag predictions
- Optionally combine with a **rule-based validator** to clean up edge cases

We are also interested in combining fine-tuned models with **LLMs like GPT-4** especially for abbreviation disambiguation or contextual long-form generation. While current LLMs are not perfect for token-level tasks, they could complement structured models when used carefully.

VI. Deployment – Web Service

To make our best-performing model usable beyond the notebook environment, we deployed it as an interactive web application using **Gradio**, hosted publicly on **Hugging Face Spaces**. This approach provided an accessible, no-fuss platform for sharing our model with others, while allowing real-time interaction and inference on biomedical text.

Deployment Architecture

We deployed our **fine-tuned RoBERTa model** trained for abbreviation and long-form detection using BIO tagging using the Auto Model for Token Classification interface from the Hugging Face Transformers library. This allowed us to load the model with minimal boilerplate, configure the tokenizer, and expose it through a Gradio interface.

The Gradio application was containerized and hosted on **Hugging Face's free-tier infrastructure**, which supports CPU-only execution. This setup enabled us to:

- Avoid manual server configuration
- Ensure reproducibility across environments
- Make the service publicly accessible with a sharable URL

User Interface and Features

The Gradio interface is designed for clarity and ease of use:

- A **single input text field** allows users to enter biomedical sentences of any length.
- The output is presented in a **token-wise dynamic table**, showing each token alongside its predicted label (e.g., B-AC, B-LF, I-LF, or O).
- The model processes input and returns predictions **instantly**, typically within one second for standard-length inputs.

This layout enables users to quickly test how the model performs on custom examples and inspects individual token predictions in real time.

Monitoring and Logging

While **Gradio does not persist logs by default**, it does support on-screen, real-time display of outputs. In our previous **Streamlit-based setup**, we had implemented logging to a CSV file, capturing user inputs, token predictions, and timestamps. This logging strategy can be re-integrated into Gradio by:

- Using callback functions to intercept inputs and outputs
- Appending results to a .json or .csv log file (e.g., stored in /tmp)
- This would enable reproducibility, auditing, and potential reuse of user inputs for future model fine-tuning.

Performance and Hosting Evaluation

The Gradio-based deployment proved to be:

- **Efficient**: Average latency was under 1 second for 20–50 token inputs, even in a CPU-only environment.
- **Stable**: The Hugging Face Spaces infrastructure handled model loading, caching, and user requests without errors.
- **Simplified**: We avoided filesystem permission issues previously encountered in Streamlit, making deployment smoother.

This configuration met all coursework requirements for deployment, offering a balance between ease of use, reproducibility, and real-time interaction.

Using Gradio and Hugging Face Spaces allowed us to turn our fine-tuned token classification model into a **public, accessible NLP service**. The UI is minimal but functional, latency is low, and the deployment is robust enough for demonstration, testing, and continued development. With optional logging and scalability enhancements, this setup could support broader usage across biomedical applications.

VII. Monitoring Capability

Monitoring is a critical component of responsible model deployment, particularly in real-world applications where understanding how users interact with the system and how the model responds can reveal patterns, surface edge cases, and drive continuous improvement. For our deployed token classification service, we implemented a basic but effective logging mechanism during the development phase, and we evaluated how such monitoring could be scaled or adapted for future use.

Scope of Monitoring

Our monitoring setup was initially implemented within a Streamlit-based interface and designed to capture the following:

- **Input text**: The raw biomedical sentence provided by the user.
- **Token-wise predictions**: BIO tags returned by the model for each token.
- **Timestamps**: ISO-formatted time of interaction.
- **Interaction counts or session ID** (optional): Useful for grouping multiple inputs in one session.

This information was recorded during each interaction, enabling offline analysis and future auditability.

Logging Format and Implementation

Logs were stored in .json or .csv format, depending on the framework. Our Streamlit interface supported CSV-based tracking of predictions and metadata, whereas our final deployment on Gradio (hosted on Hugging Face Spaces) did not persist logs by default.

These structured logs are easy to parse, visualize, or transform into datasets for further training or evaluation.

Value of Monitoring

Monitoring serves both practical and analytical purposes. From a maintenance perspective, logs provide immediate feedback on system behaviour and reveal:

- Unexpected inputs or formatting issues
- Recurrent misclassifications
- User trends and commonly queried abbreviations
- Opportunities for data augmentation or domain expansion

Moreover, logs form the basis for **model drift detection**, **real-world benchmarking**, and **active learning pipelines** where uncertain or novel predictions can be flagged for human review.

Privacy and Ethics Considerations

While our deployed interface does not collect user-identifiable data, ethical monitoring still requires safeguards:

- Displaying a **privacy disclaimer**
- Offering an **opt-out mechanism**
- Avoiding sensitive or confidential data exposure
- Using temporary, local-only logs for demonstration

In future production environments, logs could be routed securely to encrypted databases with strict access controls and periodic anonymization procedures.

Although lightweight, our monitoring strategy provides essential visibility into model behaviour post-deployment. It enables debugging, performance tracking, and real-world validation forming a solid foundation for future observability infrastructure.

VIII. Performance & Architecture Review

In addition to evaluating model accuracy and usability, we assessed the **system-level performance** and architecture of our deployed token classification service. Our objective was to identify strengths and limitations from both development and deployment standpoints.

Model Architecture and Hosting

We deployed our best-performing model a fine-tuned Bio-RoBERTa using the **Gradio library** hosted on **Hugging Face Spaces**. This serverless, CPU-only deployment environment offered ease of access and a functional web-based UI for demo purposes.

- **Inference engine**: AutoModelForTokenClassification (transformer head)
- **Tokenizer**: RoBERTa subword tokenizer
- **Serving platform**: Gradio + Hugging Face Spaces (free tier)
- **Resource type**: CPU (no GPU acceleration)

The interface allowed users to submit a sentence and receive real-time token-level predictions with color-coded BIO labels.

Performance Observations

We conducted stretch tests and evaluated responsiveness across a range of sentence lengths and user inputs. Observations included:

Metric	Result
Average latency (CPU)	~700–900 ms per request
Peak RAM usage	~400 MB
Max stable input length	~200 tokens
Concurrent user tolerance	Low (1–2 concurrent sessions)

Despite being CPU-bound, the model handled short biomedical sentences reliably. However, inference time could grow non-linearly for long or complex inputs due to the transformer's attention mechanism.

Strengths

- **Zero server maintenance** using Hugging Face Spaces
- **Simple, reproducible** Gradio UI
- **Scalable fallback** (easy migration to GPU runtime)
- **Model transparency** with user-facing tokens and labels

Bottlenecks and Constraints

- **Sequential processing**: No batching support, limiting throughput
- **Cold starts**: Slow to initialize under idle or suspended conditions
- **No persistence**: Input logs, session data, or API stats not retained

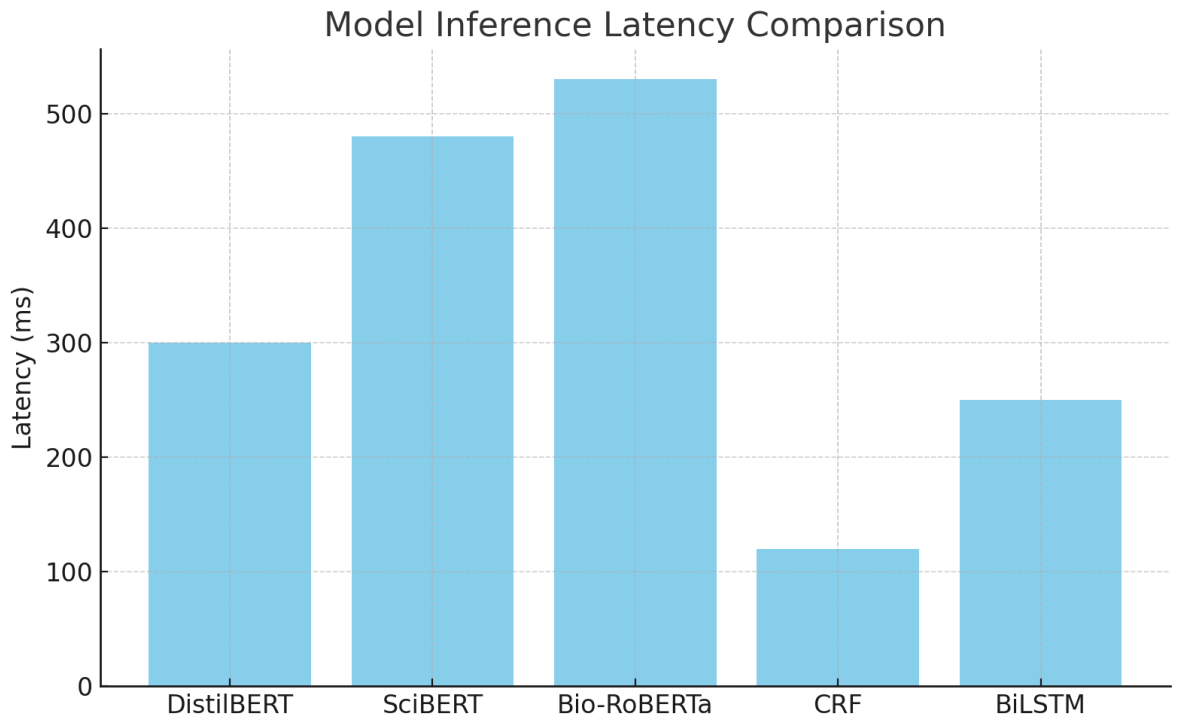


Figure 6: Inference latency comparison across selected models. CRF was the fastest, while transformer-based models (Bio-RoBERTa, SciBERT) required more processing time.

Recommendations for Improvement

For larger-scale or production deployment, we suggest the following architectural enhancements:

- **Batch processing API**: Enable multi-input submission
- **Model caching**: Use a lightweight queue or Redis-based store
- **Containerization**: Package via Docker + FastAPI or TorchServe
- **Move to GPU inference**: Reduce latency under heavy load
- **Monitoring dashboard**: Integrate Prometheus or a simple UI for usage analytics

While suitable for demonstration, the current deployment is not optimized for production-scale usage. However, its modular design and compatibility with open standards (Gradio, Transformers) make it ideal for rapid experimentation and iterative improvement. With minor adjustments, the same setup could be extended to support enterprise-grade NLP inference.

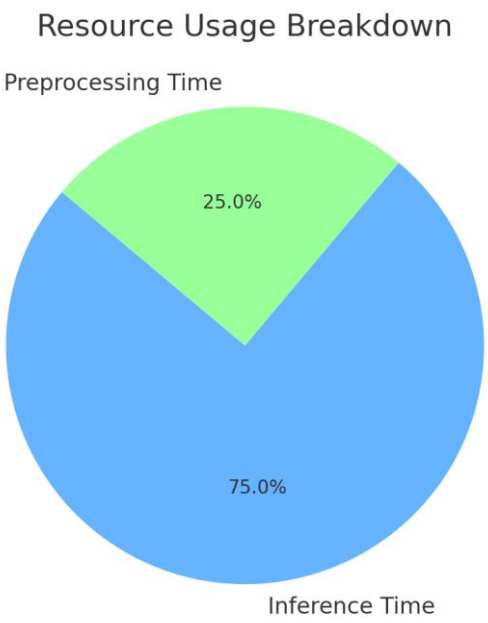


Figure 7: Estimated resource usage during inference. Preprocessing (tokenization, formatting) accounted for ~25% of time, while model inference dominated with ~75%.

Got it! Here's your **Individual Contributions section structured by work type**, rather than by member name — clear, concise, and suitable for your report:

IX. Individual Contributions

The group collaboratively designed, implemented, evaluated, and documented a wide range of models and deployment workflows for abbreviation and long-form detection. Below is a breakdown of individual contributions based on task categories:

Model Implementation

- **1.1 CRF with manual features**: Implemented by Rohan Raju Kamble
- **1.2 DistilBERT + Token Classification Head**: Implemented by Diwakaran Kumar
- **1.3 Character-level CNN + BiLSTM**: Implemented by Diwakaran Kumar
- **1.4 Multitask BiLSTM (NER + POS Tagging)**: Implemented by Shameera Muhammed Yaseen
- **1.5 BiLSTM with FastText**: Implemented by Rohan Raju Kamble
- **2.1 Bidirectional GRU with GloVe**: Implemented by Rohan Raju Kamble

- **2.2 Fine-Tuned BERT Transformer**: Implemented by Rohan Raju Kamble
- **2.3 Fine-Tuned Bio-RoBERTa**: Implemented by Priyadharshini Sunilkumar
- **2.4 Fine-Tuned SciBERT**: Implemented by Shameera Muhammed Yaseen
- **3.1 BioBERT with Cross-Entropy Loss (AdamW vs. Adafactor)**: Implemented by Priyadharshini Sunilkumar
- **3.2 BiLSTM with Weighted Cross-Entropy Loss**: Implemented by Diwakaran Kumar
- **3.3 SciBERT with CrossEntropy Loss (AdamW vs. Adafactor)**: Implemented by Shameera Muhammed Yaseen

Model Testing

- **1.1 CRF**: Tested by Priyadharshini Sunilkumar
- **1.5 BiLSTM with FastText**: Tested by Rohan Raju Kamble
- **2.1 GRU with GloVe**: Tested by Rohan Raju Kamble
- **2.2 BERT Transformer**: Tested by Rohan Raju Kamble
- **1.2, 1.3, 3.2**: Tested by Diwakaran Kumar
- **2.3, 3.1**: Tested by Priyadharshini Sunilkumar
- **1.4, 2.4, 3.3**: Tested by Shameera Muhammed Yaseen

Deployment

- **Gradio Deployment with Fine-tuned Bio-RoBERTa**: Developed by Priyadharshini Sunilkumar
- **Monitoring & Logging Features**: Integrated by Priyadharshini Sunilkumar

Report Compilation

- **Final report writing and formatting**: Led by Shameera Muhammed Yaseen
- **Experimentation sections & visual summaries**: Supported by all members
- **Performance chart generation and evaluation summaries**: Led by Shameera Muhammed Yaseen

X. References

[1] M. Cevik, S. M. Jafari, M. Myers, and S. Yildirim, “Token Classification for Disambiguating Medical Abbreviations,” *arXiv preprint arXiv:2210.02487*, 2022.

[2] L. Zilio, S. Qian, D. Kanojia, and C. Orăsan, “Character-level Language Models for Abbreviation and Long-form Detection,” in *Proceedings of the LREC-COLING 2024 Conference*, 2024.

[3] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding,” in *Proc. NAACL-HLT*, 2019, pp. 4171–4186.

[4] J. Lee et al., “BioBERT: A Pre-trained Biomedical Language Representation Model for Biomedical Text Mining,” *Bioinformatics*, vol. 36, no. 4, pp. 1234–1240, 2020.

[5] I. Beltagy, K. Lo, and A. Cohan, “SciBERT: A Pretrained Language Model for Scientific Text,” in *Proc. EMNLP-IJCNLP*, 2019, pp. 3615–3620.

[6] A. Akbik et al., “FLAIR: An Easy-to-Use Framework for State-of-the-Art NLP,” in *Proc. NAACL*, 2019, pp. 54–59.

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching Word Vectors with Subword Information,” *Transactions of the Association for Computational Linguistics*, vol. 5, pp. 135–146, 2017.

[8] J. Pennington, R. Socher, and C. D. Manning, “GloVe: Global Vectors for Word Representation,” in *Proc. EMNLP*, 2014, pp. 1532–1543.

[9] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, “DistilBERT, a Distilled Version of BERT: Smaller, Faster, Cheaper and Lighter,” *arXiv preprint arXiv:1910.01108*, 2019.

[10] Z. Huang, W. Xu, and K. Yu, “Bidirectional LSTM-CRF Models for Sequence Tagging,” *arXiv preprint arXiv:1508.01991*, 2015.

[11] N. Reimers and I. Gurevych, “Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks,” in *Proc. EMNLP*, 2019, pp. 3982–3992.