# Using character-level models for efficient abbreviation and long-form detection

**Leonardo Zilio[†1], Shenbin Qian[†2], Diptesh Kanojia[3] and Constantin Orăsan[2]**
[1]Friedrich-Alexander-Universität Erlangen-Nürnberg, Germany
[2]Centre for Translation Studies & [3]Insitute for People-Centred AI, University of Surrey, United Kingdom
leonardo.zilio@fau.de, {s.qian, d.kanojia, c.orasan}@surrey.ac.uk

**Abstract**

Abbreviations and their associated long forms are important textual elements that are present in almost every scientific communication, and having information about these forms can help improve several NLP tasks. In this paper, our aim is to fine-tune language models for automatically identifying abbreviations and long forms. We used existing datasets which are annotated with abbreviations and long forms to train and test several language models, including transformer models, character-level language models, stacking of different embeddings, and ensemble methods. Our experiments showed that it was possible to achieve state-of-the-art results by stacking RoBERTa embeddings with domain-specific embeddings. However, the analysis of our first run showed that one of the datasets had issues in the BIO annotation, which led us to propose a revised dataset. After re-training selected models on the revised dataset, results show that character-level models achieve comparable results, especially when detecting abbreviations, but both RoBERTa$_{large}$ and the stacking of embeddings presented better results on biomedical data. When tested on a different subdomain (segments extracted from computer science texts), an ensemble method proved to yield the best results for the detection of long forms, and a character-level model had the best performance in detecting abbreviations.

**Keywords:** Abbreviation detection, Character-level language models, Scientific datasets

## 1. Introduction

Abbreviations and long forms are ubiquitous in scientific communication, and their understanding can pose challenges for those who are unfamiliar with the domain, or even among peers. As such, the automatic recognition of these linguistic elements can help in Natural Language Processing (NLP) tasks that require information about these types of lexical units, such as machine translation, automatic terminology extraction, and information retrieval. As Toole (2000) mentions, in terms of information retrieval, the non-recognition of abbreviations and their long forms can lead to a negative impact on both precision and recall. In addition, a text can become completely unintelligible if abbreviations are not translated correctly, and terminology databases can be affected by the wrong automatic recognition of abbreviations and long forms.

In this paper, our aim is to train and fine-tune language models that can offer state-of-the-art (SOTA) performance in both abbreviation and long-form detection. Our work builds upon the work of Zilio et al. (2022) in abbreviation and long-form detection by expanding the evaluation to different types of pre-trained models and by conducting an extensive error analysis based on the disagreements in the language model predictions. This error analysis ultimately identified issues not only with the predicted output but also in both publicly available datasets. Our evaluation especially focuses on character-level language models' (CLMs) performance compared with standard *autoencoders* (BERT-based models), such as RoBERTa (Liu et al., 2019), and how stacked embeddings (Akbik et al., 2019) and ensembling can further improve the task performance. CLMs, which are modelled on character-level vocabulary, have achieved SOTA performance on multiple Named Entity Recognition (NER) datasets (Akbik et al., 2018). NER is an NLP task modelled as token classification, akin to abbreviation and long-form detection. This focus on CLMs is based on our *hypothesis that CLM stacking should help achieve better performance* on a task dealing with character sequences unknown to the pre-trained language model's (autoencoders) token-level vocabulary.

Our results show an improvement over SOTA for the detection of both abbreviations and long forms on the PLOD dataset (Zilio et al., 2022), which contains texts from the biomedical domain. In addition, we were able to improve over previous results on the English scientific data provided by Veyseh et al. (2022a) in the Shared Tasks at the Scientific Document Understanding Workshop 2022 (SDU dataset). The SDU dataset is composed by segments extracted from the domain of computer science, especially papers from the collection of the Association for Computational Linguistics (ACL), and fine-tuning on CLMs presented the best results for abbreviation detection on this dataset. The **key contributions** of our work can be summarised as

†Both authors contributed equally to this work.



*LREC-COLING 2024*, pages 3028–3037
20-25 May, 2024. 

follows:

- Our work achieved SOTA performance for detecting abbreviations and long forms on the PLOD and the SDU datasets using stacked embeddings and an ensemble method.
- A comprehensive evaluation was presented based on various language model combinations.
- We built pre-trained lightweight CLMs which can be fine-tuned to achieve performance comparable to Transformer-based fine-tuned models (Vaswani et al., 2017), which are significantly larger in size.
- Our approach evaluated the models' robustness in generalisation to out-of-distribution subdomains. Both SDU and PLOD datasets are from the overarching scientific domain, but PLOD has biomedical data, while SDU contains mainly extracts from computational linguistics research.
- We performed a detailed error analysis and discussed errors in model predictions and datasets.
- We re-annotated the PLOD BIO data[1] and proposed a new version of the dataset[2] that corrected BIO errors and increased the total number of abbreviations in the dataset.

In the remainder of this paper, Section 2 reviews related work for abbreviation detection with a focus on CLMs. Section 3 describes the datasets used for training and testing our proposed methods. Section 4 explains the methods we used to improve the performance of token classifiers for abbreviation detection. Section 5 shows the results of our trained models and discusses the effectiveness of our proposed methods. Section 6 concludes the paper with a summarisation of the methods and results and suggests directions for future work.

## 2. Related work

Token classification or sequence labelling is a known method for modeling foundational NLP tasks like NER (Murthy et al., 2022) and part-of-speech tagging (Chiche and Yitagesu, 2022). It is also applied to word-level quality estimation (Zerva et al., 2022), abbreviation and long-form detection (Zilio et al., 2022; Veyseh et al., 2022b). While these tasks cater to different problems in NLP, common approaches, such as fine-tuning pre-trained language models (PTLM) in a supervised setting, have helped improve the task performance for all of them. To this end, the development of frameworks, like spaCy (Honnibal et al., 2020), Stanza (Qi et al., 2020), and FLAIR (Akbik et al., 2019), along with web-based tools like MadDog (Pouran Ben Veyseh et al., 2021), helps propagate NLP research for all these tasks.

Approaches that utilise methods based on rules (Park and Byrd, 2001; Schwartz and Hearst, 2002; Kirchhoff and Turner, 2016) or feature engineering (Kuo et al., 2009; Liu et al., 2017; Li et al., 2018; Hasso et al., 2022, 2023) are outperformed by those based on deep learning, which use neural networks to achieve a better generalisation capability for detection of cross-domain, out-of-domain and multilingual abbreviations (Wu et al., 2015; Zhu et al., 2021; Li et al., 2021). Wang et al. (2023) tested the capacity of GPT3 for the task of NER on English datasets and showed that generative models have worse performance than supervised baselines.

In 2022, Zilio et al. (2022) released the largest English dataset for abbreviation and long-form detection. The efficacy of the released dataset, PLOD, was evaluated by fine-tuning several PTLM variants based on the Transformers architecture, including a variant from the Sentence-BERT (Reimers and Gurevych, 2019) architecture.

On the one hand, PTLMs use token-level information to predict masked tokens, and sentence-level context to predict the next sentence (Devlin et al., 2019). PTLMs also work with a set vocabulary of known words and a tokenisation process which helps break down unknown words and approximate their embeddings in the semantic space[3]. On the other hand, character-level language models (CLM) alleviate the issue of unknown words by predicting characters using the preceding characters as information. They utilise the hidden states of a forward-pass LSTM and a backward-pass LSTM (*i.e.,* forward-backward LM) to create contextualised word embeddings. Both forward and backward passes are pre-trained separately, and the embeddings generated from them are stacked together for further fine-tuning.

Intuitively, abbreviations are usually derived from character sequences that appear on their long forms, so, in this paper, we evaluate pre-trained CLMs, and, in addition, we stack the embeddings from Transformer-based PTLMs and word embedding models like GloVe (Pennington et al., 2014)

[1] BIO is an annotation scheme to train token classifiers, in which abbreviations were annotated as B-AC (*i.e.*, Begin Abbreviation), and tokens that were a part of long forms were assigned B-LF (*i.e.*, Begin Long Form) at the beginning, and I-LF (*i.e.*, Inside Long Form) in the middle or end. Tokens that did not belong to abbreviations or long forms were assigned B-O, which stands for Other.

[2] The dataset is available at https://github.com/surrey-nlp/PLODv2-CLM4AbbrDetection.

[3] For example, the word "sleeping" could be tokenised into "sleep" and "##ing", as "sleep" is a known word in the PTLM vocabulary.

and FastText (Bojanowski et al., 2017), performing an almost exhaustive evaluation of LM combinations for abbreviation detection. To this end, we also pre-train a CLM on the PLOS textual data, and continue the pre-training of the existing PubMed-based CLM using PLOS data, helping to push the SOTA boundary.

# 3. Datasets

## 3.1. Pre-training Dataset

For in-domain training of CLMs, we extracted the textual content from the annotated segments in PLOD dataset[4]. This resulted in a corpus of 42,889,363 tokens which we used to pre-train both forward and backward passes of a new CLM, which we refer to in this paper as the *CLM-PLOS* language model.

We also used the same data for continued pre-training of the existing CLM-PubMed language model to generate the *CLM-PubMed-PLOS* language model. We provide the details of this pre-training and continued pre-training in Section 4.

## 3.2. PLOD v1

The original PLOD dataset (which we call PLOD v1) (Zilio et al., 2022) is currently the largest dataset for abbreviation and long-form detection. It is available in two variants: an *unfiltered* and a *filtered* dataset. The difference between these two variants is not significant in terms of numbers, as both contain more than 160k annotated segments. Still, the filtered variant went through semi-automatic validation of the annotation, so it should be a more reliable resource.

The annotation was based on manually constructed glossaries of published papers, which are available in some articles of the PLOS Journals[5]. The manually constructed glossaries in each paper were used as input for automatically detecting abbreviations and long forms. Given the automatic nature of the annotation, it is expected that this dataset will present some issues, some of which have also been pointed out by the authors of the dataset.

Both variants (filtered and unfiltered) of the PLOD dataset are available in tab-separated values (TSV) format, as a single file, and for comparability purposes, as separate BIO-annotated train, validation and test sets. We used the BIO-annotated files to fine-tune and test our models. We also used the TSV files to extract the raw textual information from the PLOS journals for training character-level language models, as mentioned in Section 3.1.

[4]The full, non-tokenised textual content of the segments is available as a TSV file in PLOD's Github: https://github.com/surrey-nlp/PLOD-AbbreviationDetection.

[5]The PLOS Journals are available at https://plos.org/.

### 3.2.1. Issues with PLOD v1 BIO annotation

After training our models on the unfiltered PLOD v1, during the error analysis stage, we analysed 300 random segments of annotated data (which contained 1,527 annotated tokens), and we realised that the tokens in the BIO datasets were not actually matching the segments available in the TSV files of PLOD; an error that was probably originated when the data in TSV format was converted to BIO format, as the TSV files were checked without finding any of the issues discussed here.

There were mainly three BIO annotation issues: **1)** misplacement of abbreviation tokens; **2)** duplication of abbreviation tokens; and **3)** misannotation of tokens. For example, the following sequence is part of a BIO-annotated segment in the unfiltered PLOD v1 (each line is a triplet composed of token, part-of-speech tag, and BIO annotation):

| | | |
|---|---|---|
| positron | NOUN | B-LF |
| FP-CIT | PROPN | B-AC |
| emission | NOUN | I-LF |
| tomography | NOUN | I-LF |
| ( | PUNCT | B-O |
| PET | PROPN | B-AC |
| ) | PUNCT | B-O |

For some reason, the token "FP-CIT" was misplaced right in the middle of the long-form "positron emission tomography", while the original segment presents the following order:

"FP-CIT positron emission tomography (PET)"

These issues led us to reconsider using the currently available BIO data, which was kept in this paper just for the sake of comparability with the models trained by Zilio et al. (2022). After this, we created a new BIO conversion script based on the PLOD dataset TSV files. We describe this process and its outcome in the next subsection.

## 3.3. PLOD v2 and SDU

We used the indices of abbreviations and long forms in the TSV files of the PLOD data described in Section 3.2 and re-created the BIO dataset by adjusting the indices to token boundaries. As shown in Table 1, the BIO re-annotation process not only improved the accuracy of BIO labels but also increased the number of abbreviations while keeping almost the same amount of long forms. After the re-annotation, the number of abbreviations in the filtered and unfiltered variants increased by almost 40k instances[6]. The re-annotated BIO dataset (PLOD v2) was randomly split into 70% of the segments for training, 15% for development, and 15%

[6]The code for this re-annotation and the re-annotated BIO data is available along with our models in our GitHub repository: https://github.com/surrey-nlp/PLODv2-CLM4AbbrDetection.

for testing. These splits were used for fine-tuning token classification models.

| | PLOD v1 | | PLOD v2 | |
|---|---|---|---|---|
| | AC | LF | AC | LF |
| *Filtered* | 439,447 | 262,745 | 478,275 | 261,721 |
| *Unfiltered* | 448,090 | 266,534 | 486,956 | 265,466 |

Table 1: The number of abbreviations (AC) and long forms (LF) before and after re-annotation.

To test the generalisation ability of trained models and compare with results in the PLOD paper (Zilio et al., 2022), we used the English dataset for abbreviation detection from the SDU dataset provided by Veyseh et al. (2022a). Following the same procedure applied by Zilio et al. (2022), we combined both the train and validation sets released by the shared-task organisers into a single test set. Since the SDU data are not BIO annotated (provided with raw texts and indices of abbreviations and long forms), we re-annotated the data using our new BIO converter to get the correct annotations.
The number of abbreviations and long forms in the re-annotated data used for training, validation and testing is shown in Table 2.

| | | AC | LF |
|---|---|---|---|
| **PLOD v2** *filtered* | train | 334294 | 183347 |
| | valid | 71838 | 39247 |
| | test | 72143 | 39127 |
| **PLOD v2** *unfiltered* | train | 341894 | 186402 |
| | valid | 72888 | 39877 |
| | test | 72174 | 39187 |
| **SDU** | train+valid | 9167 | 6411 |

Table 2: The number of abbreviations (AC) and long-forms (LF) in re-annotated PLOD v2 and SDU datasets.

## 4. Methodology

To improve the performance of abbreviation and long-form detection algorithms, we explored different methods, namely, fine-tuning existing CLMs, pre-training new CLMs, continued pre-training of existing CLMs, and stacked embedding models. We fine-tuned our token classifiers on PTLMs, existing CLMs and resultant CLMs from this work, while also stacking embeddings from two or more language models. In addition, we used an ensemble method to observe whether pooling model predictions together would yield better results.

### 4.1. Base models

**Existing CLMs.** We used both in-domain and out-of-domain models, *i.e.,* FLAIR-News language model (CLM-News) and PubMed language model (CLM-PubMed), released by Akbik et al. (2018). These CLMs were all trained on sequences of characters without any explicit notion of words/tokens in their vocabulary. CLM-News was trained with a 1-billion-word corpus, and CLM-PubMed was trained with 5% of PubMed (biomedical) abstracts.

**Pre-training and continued pre-training.** We used in-domain PLOS Journals to first pre-train a CLM using the FLAIR framework[7] (Akbik et al., 2019), which generated the CLM-PLOS model. After that, we used the same PLOS data for a continued pre-training of the existing CLM-PubMed model, which generated the CLM-PubMed-PLOS model. For both pre-training and continued pre-training, we set the non-annealed *learning rate* to 20, the *hidden size* to 2048, the number of *layers* to 1, the *sequence length* to 256, the *batch size* to 200, and the number of *epochs* to 300. Pre-training of CLMs was performed on a single NVIDIA A40 for about 14 days (7 days per pass). Continued pre-training was carried out on a single NVIDIA RTX A5000, which also took about 14 days.

**Stacking of embeddings.** We tested different concatenations of embeddings using FLAIR (Akbik et al., 2018), including CLM-PubMed embeddings, embeddings from our pre-trained CLM-PLOS and continually pre-trained CLM-PubMed-PLOS, in combinations with Transformer-based embeddings from BERT (Devlin et al., 2019) and RoBERTa (Liu et al., 2019), and with word embeddings from GloVe and FastText. The final sequence representation is given, for example, by:

$$W_i = \begin{bmatrix} W_i^{GloVe} \\ W_i^{CLM-PLOS} \end{bmatrix}$$

Here, $W_i^{GloVe}$ is the GloVe embedding (Pennington et al., 2014), and $W_i^{CLM-PLOS}$ is the embedding from our pre-trained character-level language model on the PLOS dataset.

### 4.2. Fine-tuning for token classifiers

After preparing the base models, the second step was submitting them to fine-tuning in order to get token classifiers for abbreviation and long-form detection. All these base models were first fine-tuned on the misannotated unfiltered PLOD v1 BIO data to: 1) investigate which method would give us the best result; and 2) to obtain results comparable to those in the release paper of the PLOD dataset. After testing which methods would lead to better results, we used the best models to fine-tune on the re-annotated PLOD v2 BIO data for improving upon the state of the art.

**Training details.** All base models were fine-tuned using the FLAIR framework with default hyperparameters. We set the *learning rate* to 0.01,

[7]For more information: https://github.com/flairNLP/flair.

and the number of *epochs* to 150; we also opted for a linear scheduler with *warmup* to slowly decrease the learning rate. The *batch size* for fine-tuning most models was 32, except for the models that stacked RoBERTa embeddings, for which we set it to 2, in accordance with the capacity of our GPU. To compare with the state of the art provided in Zilio et al. (2022), we also fine-tuned the RoBERTa$_{large}$ model using the Jupyter notebook on the PLOD GitHub without changing any of the hyperparameters. The overall fine-tuning experiments for this work took about 3 months. Each resultant model which used stacked RoBERTa$_{large}$ embeddings took approximately 4 weeks to train on the unfiltered variant, and 3 weeks on the filtered variant of PLOD[8].

### 4.3. Ensemble method

After having fine-tuned all the models, we applied an ensemble method to identify whether the results would improve over individual models. For this, we used majority voting over the predictions obtained from the top three models. Tie-breakers (for the cases in which all three systems disagreed) were decided in favour of the model that presented the individual best predictions on the test sets. This best model was selected based on an unweighted average of F1 scores in abbreviation and in long-form detection.

## 5. Results and Discussion

### 5.1. Model results

Table 3 shows the results of models fine-tuned on misannotated, unfiltered PLOD v1 BIO data. We treat existing results from the PLOD paper as baseline results (top two rows in the table). The third and fourth rows show results on fine-tuning existing CLMs trained on out-of-domain news data (CLM-News) and in-domain PubMed data (CLM-PubMed). We observe that both results are significantly better than the baseline ALBERT$_{base}$ model, which suggests that they can achieve comparable results to BERT-like transformer models. The fifth and sixth rows in Table 3 show the performance of our pre-trained CLMs on the PLOS data (CLM-PLOS) and continually pre-trained PubMed model with the PLOS data (CLM-PubMed-PLOS). We can see that, as expected, in-domain pre-training consistently improves the CLM performance, while continued pre-training only slightly improves the base model's performance in detecting abbreviations. Although the results for these CLMs is still below the performance of the large variant of the RoBERTa model, they are significantly lighter in size even when combining both passes of the CLMs.

From the seventh to the last row in Table 3, models were fine-tuned with combinations of different embeddings. We used stacked embeddings of GloVe and CLM-PubMed, GloVe and CLM-PLOS, FastText and CLM-PLOS, CLM-PubMed and CLM-PLOS. In addition, we also stacked embedding combinations from BERT and CLM-PLOS, RoBERTa$_{large}$ and CLM-PubMed-PLOS; FastText, RoBERTa$_{large}$ and CLM-PubMed-PLOS; and, finally, GloVe, RoBERTa$_{large}$ and CLM-PubMed-PLOS. We observe from the stacked embeddings models that simply stacking more embeddings does not help the performance. When we stacked word embeddings like FastText and GloVe with transformer embeddings, they seemed to add noise to the training, and the results were no better than without them.

We got our new SOTA from the stacked embeddings model of RoBERTa$_{large}$ plus CLM-PubMed-PLOS, which achieved the highest F1 score in both abbreviations and long forms.

As we discussed in Section 3, PLOD v1 has issues related to the BIO annotation. Hence, the next two tables present the results on re-annotated PLOD v2 BIO data, including selected models from our initial investigation and results from the ensemble method we experimented with. We left the SOTA from Zilio et al. (2022) at the top of each table just for a quick reference, *but the datasets are now different, as we re-annotated the BIO files and used new random splits*.

Table 4 shows the results of models fine-tuned on re-annotated, **unfiltered** PLOD v2 BIO data. We fine-tuned one RoBERTa$_{large}$ model, one PubMed model (CLM-PubMed), which is a character-level language model, and one stacked embeddings model, which is composed of embeddings from RoBERTa$_{large}$ and from the continued pre-trained model based on the PLOS data (CLM-PubMed-PLOS). On this dataset, we achieved the highest F1 scores using RoBERTa$_{large}$ stacked with CLM-PubMed-PLOS embeddings for abbreviation detection. This is the same result as we had in PLOD v1. However, for detecting long forms, the majority voting further improved the results by more than one percentage point (which is a large margin, considering that we have a large test set, with more than 100k annotated tokens).

When performing predictions on the SDU dataset, which contains document from different scientific subdomains, RoBERTa-based models did not achieve the best results in either abbreviation or long form detection. In abbreviation detection, CLM-PubMed showed a much higher recall performance than the other models, which pushed it to achieve the best F1 score. In terms of long form detection, the ensemble method drastically improved the performance compared to individual models. This

[8]The FLAIR version we used employs SGD optimizer which is very slow compared to the more recent AdamW.

| Pre-trained Language Model (PTLM) | Abbreviations | | | Long Forms | | |
|---|---|---|---|---|---|---|
| | P | R | F | P | R | F |
| ***Baselines*** | | | | | | |
| *ALBERT$_{base}$ (baseline on PLOD)* | *0.8450* | *0.8980* | *0.8710* | *0.7580* | *0.8120* | *0.7840* |
| *RoBERTa$_{large}$ (SOTA on PLOD)* | *0.9110* | *0.9350* | *0.9220* | *0.8760* | *0.9210* | *0.8980* |
| ***Existing CLMs*** | | | | | | |
| CLM-News | 0.8604 | 0.9171 | 0.8879 | 0.8190 | 0.8624 | 0.8402 |
| CLM-PubMed | 0.8778 | 0.9245 | 0.9005 | 0.8438 | 0.8860 | 0.8644 |
| ***CLMs Pre-trained w/ PLOD train data*** | | | | | | |
| CLM-PLOS | 0.8722 | 0.9284 | 0.8994 | 0.8326 | 0.8782 | 0.8548 |
| CLM-PubMed-PLOS | 0.8761 | 0.9279 | 0.9013 | 0.8362 | 0.8879 | 0.8613 |
| ***Stacked Embeddings*** | | | | | | |
| GloVe + CLM-PubMed | 0.8729 | 0.9272 | 0.8992 | 0.8368 | 0.8830 | 0.8593 |
| GloVe + CLM-PLOS | 0.8750 | 0.9295 | 0.9015 | 0.8365 | 0.8795 | 0.8575 |
| FastText + CLM-PLOS | 0.8782 | 0.9312 | 0.9039 | 0.8396 | 0.8817 | 0.8601 |
| CLM-PubMed + CLM-PLOS | 0.8811 | 0.9333 | 0.9065 | 0.8398 | 0.8828 | 0.8608 |
| BERT + CLM-PLOS | 0.8978 | 0.9338 | 0.9154 | 0.8465 | 0.8991 | 0.8720 |
| RoBERTa$_{large}$ + CLM-PubMed-PLOS | **0.9164** | **0.9456** | **0.9308** | 0.8833 | **0.9308** | **0.9065** |
| FastText + RoBERTa$_{large}$ + CLM-PubMed-PLOS | 0.9145 | 0.9413 | 0.9277 | **0.8841** | 0.9263 | 0.9047 |
| GloVe + RoBERTa$_{large}$ + CLM-PubMed-PLOS | 0.8800 | 0.9317 | 0.9051 | 0.8316 | 0.8959 | 0.8625 |

Table 3: Abbreviation Detection performance using various language model combinations evaluated using Precision (P), Recall (R), and F1-score (F), trained on the unfiltered PLOD v1 BIO data.

goes to show that both CLM and the ensemble method are more robust in cross-domain scenarios.

Table 5 shows the results of models fine-tuned on re-annotated, filtered PLOD v2 BIO data. We can see that RoBERTa$_{large}$ is the best model in terms of detecting abbreviations in the PLOD v2 test set, but it is surpassed by the majority voting ensemble method when it comes to long-form detection. Here it is important to highlight that, for abbreviation detection, the ensemble method is only slightly worse (0.1 point) in terms of F1 score, as it has better recall than RoBERTa$_{large}$. So, on PLOD v2, the ensemble method achieves the best overall performance for the task of abbreviation and long-form detection. When tested on the SDU dataset, this becomes clearer, as RoBERTa$_{large}$ does not perform well at detecting either abbreviations or long forms (similar to what happened with the unfiltered dataset). The best results on SDU were achieved using the CLM-PubMed for abbreviation detection, and the ensemble method for long-form detection, again proving that the CLM offers better recall than the other models.

The re-annotation of the BIO data and the re-training of these models not only improved the performance on the PLOD test set, but also enhanced the generalisation ability of the models, as the F1 scores see a huge boost on the SDU dataset when compared with the results presented in the PLOD paper (Zilio et al., 2022).

Overall, considering our interest in comparing character-level language models with transformer models, our results revealed that character-level models can achieve comparable results in the detection of abbreviations, where they can even surpass transformer models, especially in terms of generalisation. We also noticed that CLMs usually improve the recall of predictions on the test sets. However, while they performed well on abbreviation detection, they were much less reliable for long-form detection, where they were consistently outperformed by large Transformer models. The stacking of different embeddings, especially embeddings from large Transformer models proved to be robust in both abbreviation and long-form detection, being surpassed only in specific scenarios of our experiments, and even so, achieving comparable results. Using an ensemble of different models, even as a simple majority voting system, proved useful to further improve model performance when it comes to long-form detection. This was especially true in the test of generalisation, when the models were applied to the SDU dataset. In this case, the results of the majority voting were around 8 percentage points higher than the second best model.

## 5.2. Disagreement analysis

After running our three selected models on both the re-annotated, **filtered** PLOD v2 test set and the SDU dataset, we analysed which tokens were causing disagreements in the output. We randomly selected a sample of 300 segments from the filtered PLOD v2 test set and another 300 from the SDU dataset in which at least one token presented disagreement either among the three systems or between the models and the test sets. As such, *we only looked at tokens in which either one of the*

| Language Model | PLOD Test *unfiltered* | | | | | | SDU (Train+Dev Set) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Abbreviations | | | Long Forms | | | Abbreviations | | | Long Forms | | |
| | P | R | F | P | R | F | P | R | F | P | R | F |
| *SOTA on PLOD unfiltered* | *0.9110* | *0.9350* | *0.9220* | *0.8760* | *0.9210* | *0.8980* | *0.7070* | *0.6410* | *0.6720* | *0.4230* | *0.1910* | *0.2640* |
| RoBERTa$_{large}$ | 0.8916 | 0.9152 | 0.9033 | 0.8607 | 0.9142 | 0.8867 | 0.9075 | 0.8115 | 0.8568 | 0.8029 | 0.7149 | 0.7564 |
| CLM-PubMed | 0.8502 | 0.9155 | 0.8816 | 0.8270 | 0.8864 | 0.8557 | 0.8941 | **0.8845** | **0.8893** | 0.7594 | 0.6411 | 0.6953 |
| RoBERTa$_{large}$ + CLM-PubMed-PLOS | **0.8977** | 0.9351 | **0.9160** | 0.8726 | 0.9260 | 0.8985 | 0.9012 | 0.8207 | 0.8590 | 0.7896 | 0.7348 | 0.7613 |
| Ensemble | 0.8893 | **0.9362** | 0.9121 | **0.8758** | **0.9482** | **0.9106** | **0.9148** | 0.8574 | 0.8852 | **0.8449** | **0.8524** | **0.8487** |

Table 4: Abbreviation Detection performance using various language model combinations evaluated using Precision (P), Recall (R), and F1-score (F), **trained on re-annotated, *unfiltered* PLOD v2 BIO data**, and tested on both.

| Language Model | PLOD Test *filtered* | | | | | | SDU (Train+Dev Set) | | | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | Abbreviations | | | Long Forms | | | Abbreviations | | | Long Forms | | |
| | P | R | F | P | R | F | P | R | F | P | R | F |
| *SOTA on PLOD filtered* | *0.9060* | *0.9350* | *0.9200* | *0.8740* | *0.9250* | *0.8980* | *0.7280* | *0.6430* | *0.6830* | *0.5200* | *0.1690* | *0.2550* |
| RoBERTa$_{large}$ | **0.9073** | 0.9348 | **0.9208** | **0.8908** | 0.9318 | 0.9108 | 0.9155 | 0.8074 | 0.8580 | 0.8074 | 0.7197 | 0.7610 |
| CLM-PubMed | 0.8467 | 0.9226 | 0.8830 | 0.8185 | 0.8887 | 0.8522 | 0.9117 | **0.8708** | **0.8908** | 0.7650 | 0.6464 | 0.7007 |
| RoBERTa$_{large}$ + CLM-PubMed-PLOS | 0.8924 | 0.9375 | 0.9144 | 0.8750 | 0.9225 | 0.8981 | 0.9162 | 0.8238 | 0.8675 | 0.7799 | 0.7245 | 0.7512 |
| Ensemble | 0.8946 | **0.9464** | 0.9198 | 0.8872 | **0.9529** | **0.9189** | **0.9256** | 0.8500 | 0.8862 | **0.8395** | **0.8521** | **0.8457** |

Table 5: Abbreviation Detection performance using various language model combinations evaluated using Precision (P), Recall (R), and F1-score (F), **trained on re-annotated, *filtered* PLOD v2 BIO data**, and tested on both.

*models or the test set annotation disagreed with the other models*.

The error analysis that we report here was purposefully based on disagreements. We expected that the points of disagreement would represent difficult cases in the datasets, which could also indicate cases of misannotation. So it is very important to keep in mind that we were not just looking at vanilla annotations in the datasets. Since such type of analysis requires a higher level of linguistic training, it was conducted by a linguist.

Tables 6 and 7 present the results of the disagreement analysis conducted on the filtered PLOD v2 test set. It can be seen that, whenever the test set disagrees with the systems (meaning that all three models agree on the annotation), it is usually a mistake on the test set annotation, as it presented issues in almost 73% of the instances. Of all three models, RoBERTa$_{large}$ seems to be the one that was more influenced by the errors in the dataset, as there were errors in the test set on almost 51% of the time in which RoBERTa$_{large}$ was wrong in disagreeing with the other two models. This percentage was much lower for the other two models. In this disagreement analysis, CLM-PubMed was actually right in disagreeing with the other systems around 46% of the time. As such, although RoBERTa$_{large}$ did achieve good overall results in the experiments, it seems to be far more overfitted than CLM-PubMed, confirming a point that was observed when we tested our models on the SDU dataset. Overall, *and emphasising again that we were focusing on difficult annotation cases, where there were disagreements*, the test set presented a good amount of problems, with more than 47% of the cases being erroneous, especially in terms of missing annotations (recall), which amounted to 95% of the observed errors. Because the annotation of PLOD was based on manually constructed glossaries, this lack of recall actually was an expected issue. An example of lack of annotation in PLOD can be seen in the following extract (ID 1110868 in the filtered TSV file):

> ACE/ARB, angiotensin-converting enzyme inhibitors/angiotensin receptor blockers

where there are no annotations in the dataset. “ACE” and “ARB” are not annotated as abbreviations, and, as a consequence, neither are their long forms, even though they are clearly related and should have been all annotated.

| | Disagrees w/ other models | Is wrong (%) | Issues w/ test set (%) |
|---|---|---|---|
| **Filtered PLOD v2** | 428 | 312 (72.90%) | N/A |
| **RoBERTa$_{large}$** | 384 | 282 (73.44%) | 143 (50.71%) |
| **CLM-PubMed** | 535 | 288 (53.83%) | 57 (19.79%) |
| **RoBERTa$_{large}$ + CLM-PubMed-PLOS** | 435 | 300 (68.97%) | 71 (23.67%) |

Table 6: Error analysis of disagreements on the PLOD v2 test set.

| Total Tokens | Total Errors | Precision | Recall |
|---|---|---|---|
| 1782 | 848 (47.59%) | 42 (4.95%) | 806 (95.05%) |

Table 7: Errors in the PLOD v2 test set.

Tables 8 and 9 present the results of the error analysis conducted on the SDU dataset. This analysis

was conducted using the same methodology as in the PLOD v2 dataset, so we are again looking at disagreements. In terms of model behaviour, the results on this dataset were very similar, as again RoBERTa$_{large}$ was the model that was actually wrong most of the time when disagreeing with the other two models. The SDU dataset has less annotation errors, but it still was more wrong (53%) than right when disagreeing with the three models. CLM-PubMed was again the best in disagreeing, as it was actually right in almost 40% of these cases. Considering the sample extracted for this error analysis, the SDU dataset presented around 34% of incorrectly annotated instances, of which almost 22% were cases of misannotation (precision). As an example of misannotation, we have the following extract (ID 644 in the JSON train file):

> 4Reviews with **NDr** = **NTr** *are regarded as incorrectly* classified by TopicSpam.

where "NDr" and "NTr" are both correctly annotated as abbreviations, but then "are regarded as incorrectly" is incorrectly annotated as a long form.

| | Disagrees w/ other models | Is wrong (%) | Issues w/ test set (%) |
|---|---|---|---|
| **SDU** | 355 | 189 (53.24%) | N/A |
| **RoBERTa$_{large}$** | 311 | 214 (68.81%) | 45 (21.03%) |
| **CLM-PubMed** | 471 | 285 (60.51%) | 55 (19.30%) |
| **RoBERTa$_{large}$ + CLM-PubMed-PLOS** | 387 | 260 (67.18%) | 19 (07.31%) |

Table 8: Error analysis of disagreements on the SDU dataset.

| Total Tokens | Total Errors | Precision | Recall |
|---|---|---|---|
| 1524 | 519 (34.06%) | 113 (21.77%) | 406 (78.23%) |

Table 9: Errors in the SDU dataset.

However, one of the main issues with the SDU dataset is actually tokenisation. On many segments, there are single characters tokenised separately from the rest of the word, or words split into several tokens. For instance, segment 193 of the JSON train file is "1 25 2. Loca l Word Grouper (LWG) The funct ion of th i s b lock is to fo rm " (sic). Not only it is an incomplete sentence, which can be problematic if a long form is broken in the middle, but the tokenisation is also completely broken, and it is, for instance, breaking one of the words in the long form "Local Word Grouper".

As we mentioned previously, by looking only at disagreements, we were probably only looking at the most complicated instances in both datasets, so it is expected that a certain amount of issues with the test sets would appear.

## 6. Conclusion

In this paper, we trained language models that offer SOTA performance in abbreviation and long-form detection. We tested existing CLMs, pre-training and continued pre-training of CLMs, and stacked embeddings models as base models, which we used for fine-tuning token classifiers. We got SOTA performance based on the stacking of different embeddings and ensemble methods. We also performed a detailed error analysis of our trained models and summarised the errors in model predictions and in both datasets. We re-annotated the PLOD BIO data and proposed a new version of the BIO-annotated PLOD data that corrected errors in the dataset and increased the total number of annotated tokens.

More than 30% of the investigated instances presented issues in both test sets. Although we were purposefully looking at only disagreements, which might be the most complicated cases in the datasets, it is hard not to put a question mark in terms of the evaluation of which models are actually performing better. At the same time, we did use the only two datasets (to our knowledge) that are currently available for abbreviation detection in English, so there is not a bulletproof dataset that could be used instead of these. The comprehensive error analysis that we conducted consists in an important contribution to the field, as it raises questions regarding the existing abbreviation datasets and opens up paths for a broader discussion regarding evaluation.

A fully manual evaluation could be the best way forward to identify the best models, but it would prove to be very costly in terms of time and effort, and it might still prove to be unreliable, if low or no agreement can be observed between evaluators. As such, we plan to conduct future investigations regarding how to better evaluate the reliability of the existing datasets, including experiments with manual evaluation in both abbreviation and long-form detection.

## 7. Limitations

In virtue of the high computing costs involved in training the models, we had to make a selection of models for re-training on PLOD v2 data, and, as such, we don't have the results for all models, as we have on the misannotated PLOD v1. For the same reason of computing cost, we also only used default hyperparameters for some of the trained models.

As we identified during the disagreement analysis, both datasets present issues in terms of precision and recall, which have an impact on the models, and on the results presented in the paper. A manual revision of either dataset, however, is not feasible within the scope of this research.

The disagreement analysis, which included observations regarding the annotation of the datasets, was carried out only by one linguist, so we do not have data on inter-rater agreement. This is something we plan to explore in the future though.

## 8. Bibliographical References


Alan Akbik, Tanja Bergmann, Duncan Blythe, Kashif Rasul, Stefan Schweter, and Roland Vollgraf. 2019. FLAIR: An easy-to-use framework for state-of-the-art NLP. In *NAACL 2019, 2019 Annual Conference of the North American Chapter of the Association for Computational Linguistics (Demonstrations)*, pages 54–59.

Alan Akbik, Duncan Blythe, and Roland Vollgraf. 2018. Contextual string embeddings for sequence labeling. In *COLING 2018, 27th International Conference on Computational Linguistics*, pages 1638–1649.

Piotr Bojanowski, Edouard Grave, Armand Joulin, and Tomas Mikolov. 2017. Enriching word vectors with subword information.

Alebachew Chiche and Betselot Yitagesu. 2022. Part of speech tagging: a systematic review of deep learning and machine learning approaches. *Journal of Big Data*, 9(1):1–25.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota. Association for Computational Linguistics.

Hussein Hasso, Katharina Großer, Iliass Aymaz, Hanna Geppert, and Jan Jürjens. 2022. Abbreviation-expansion pair detection for glossary term extraction. In *Requirements Engineering: Foundation for Software Quality: 28th International Working Conference, REFSQ 2022, Birmingham, UK, March 21–24, 2022, Proceedings*, pages 63–78. Springer.

Hussein Hasso, Katharina Großer, Iliass Aymaz, Hanna Geppert, and Jan Jürjens. 2023. Enhanced abbreviation–expansion pair detection for glossary term extraction. *Information and Software Technology*, 159:107203.

Matthew Honnibal, Ines Montani, Sofie Van Landeghem, Adriane Boyd, et al. 2020. spacy: Industrial-strength natural language processing in python.

Katrin Kirchhoff and Anne M. Turner. 2016. Unsupervised resolution of acronyms and abbreviations in nursing notes using document-level context models. In *Proceedings of the Seventh International Workshop on Health Text Mining and Information Analysis*, pages 52–60, Auxtin, TX. Association for Computational Linguistics.

Cheng-Ju Kuo, Maurice HT Ling, Kuan-Ting Lin, and Chun-Nan Hsu. 2009. Bioadi: a machine learning approach to identifying abbreviations and definitions in biological literature. In *BMC bioinformatics*, volume 10, pages 1–10. BioMed Central.

Feng Li, Zhensheng Mai, Wuhe Zou, Wenjie Ou, Xiaolei Qin, Yue Lin, and Weidong Zhang. 2021. Systems at sdu-2021 task 1: Transformers for sentence level sequence label. In *SDU@ AAAI*.

Yang Li, Bo Zhao, Ariel Fuxman, and Fangbo Tao. 2018. Guess me if you can: Acronym disambiguation for enterprises. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1308–1317, Melbourne, Australia. Association for Computational Linguistics.

Jie Liu, Caihua Liu, and Yalou Huang. 2017. Multi-granularity sequence labeling model for acronym expansion identification. *Information Sciences*, 378:462–474.

Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. 2019. RoBERTa: A Robustly Optimized BERT Pretraining Approach. *arXiv preprint*.

Rudra Murthy, Pallab Bhattacharjee, Rahul Sharnagat, Jyotsana Khatri1, Diptesh Kanojia, and Pushpak Bhattacharyya. 2022. HiNER: A Large Hindi Named Entity Recognition Dataset. In *Proceedings of the Language Resources and Evaluation Conference*, pages 4467–4476, Marseille, France. European Language Resources Association.

Youngja Park and Roy J. Byrd. 2001. Hybrid text mining for finding abbreviations and their definitions. In *Proceedings of the 2001 Conference on Empirical Methods in Natural Language Processing*.

Jeffrey Pennington, Richard Socher, and Christopher Manning. 2014. GloVe: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Amir Pouran Ben Veyseh, Franck Dernoncourt, Walter Chang, and Thien Huu Nguyen. 2021. MadDog: A web-based system for acronym identification and disambiguation. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, pages 160–167, Online. Association for Computational Linguistics.

Peng Qi, Yuhao Zhang, Yuhui Zhang, Jason Bolton, and Christopher D. Manning. 2020. Stanza: A Python natural language processing toolkit for many human languages. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*.

Nils Reimers and Iryna Gurevych. 2019. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Ariel S Schwartz and Marti A Hearst. 2002. A simple algorithm for identifying abbreviation definitions in biomedical text. In *Biocomputing 2003*, pages 451–462. World Scientific.

Janine Toole. 2000. A hybrid approach to the identification and expansion of abbreviations. In *RIAO*, pages 725–736. Citeseer.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. *Advances in neural information processing systems*, 30.

Amir Pouran Ben Veyseh, Nicole Meister, Franck Dernoncourt, and Thien Huu Nguyen. 2022a. Acronym extraction and acronym disambiguation shared tasks at the scientific document understanding workshop 2022. In *SDU@AAAI*.

Amir Pouran Ben Veyseh, Nicole Meister, Seunghyun Yoon, Rajiv Jain, Franck Dernoncourt, and Thien Huu Nguyen. 2022b. MACRONYM: A Large-Scale Dataset for Multilingual and Multi-Domain Acronym Extraction. In *arXiv*.

Shuhe Wang, Xiaofei Sun, Xiaoya Li, Rongbin Ouyang, Fei Wu, Tianwei Zhang, Jiwei Li, and Guoyin Wang. 2023. Gpt-ner: Named entity recognition via large language models. *arXiv e-prints*, pages arXiv–2304.

Yonghui Wu, Jun Xu, Yaoyun Zhang, and Hua Xu. 2015. Clinical abbreviation disambiguation using neural word embeddings. In *Proceedings of BioNLP 15*, pages 171–176, Beijing, China. Association for Computational Linguistics.

Chrysoula Zerva, Frédéric Blain, Ricardo Rei, Piyawat Lertvittayakumjorn, José G. C. de Souza, Steffen Eger, Diptesh Kanojia, Duarte Alves, Constantin Orăsan, Marina Fomicheva, André F. T. Martins, and Lucia Specia. 2022. Findings of the WMT 2022 shared task on quality estimation. In *Proceedings of the Seventh Conference on Machine Translation (WMT)*, pages 69–99, Abu Dhabi, United Arab Emirates (Hybrid). Association for Computational Linguistics.

Danqing Zhu, Wangli Lin, Yang Zhang, Qiwei Zhong, Guanxiong Zeng, Weilin Wu, and Jiayu Tang. 2021. At-bert: Adversarial training bert for acronym identification winning solution for sdu@ aaai-21.

Leonardo Zilio, Hadeel Saadany, Prashant Sharma, Diptesh Kanojia, and Constantin Orasan. 2022. Plod: An abbreviation detection dataset for scientific documents. In *Proceedings of the Language Resources and Evaluation Conference*, pages 680–688, Marseille, France. European Language Resources Association.