DCU at SemEval-2023 Task 10: A Comparative Analysis of Encoder-only and Decoder-only Language Models with Insights into Interpretability

Kanishk Verma and **Kolawole Adebayo** and **Joachim Wagner** and **Brian Davis**
ADAPT Centre, School of Computing, Dublin City University,
Dublin Ireland
{firstname.lastname}@adaptcentre.ie

Abstract

We conduct a comparison of pre-trained encoder-only and decoder-only language models with and without continued pre-training, to detect online sexism. Our fine-tuning-based classifier system achieved the 16th rank in the SemEval 2023 Shared Task 10 Subtask A that asks to distinguish sexist and non-sexist texts. Additionally, we conduct experiments aimed at enhancing the interpretability of systems designed to detect online sexism. Our findings provide insights into the features and decision-making processes underlying our classifier system, thereby contributing to a broader effort to develop explainable AI models to detect online sexism.

1 Introduction

Sexism represents *"assumptions, beliefs, theories, stereotypes, and broader cultural narratives that represent men and women as importantly different"* (Manne, 2017). Sexist beliefs construct and strengthen narratives that normalise gender distinctions between males and females by promoting the idea of innate variances between the two genders. According to feminist studies (Drakett et al., 2018; Andreasen, 2021), occurrences of online sexism and harassment are frequently depicted as "permissible" by presenting them as a kind of amusement or joke. Challenging these narratives and working to eliminate gender-based discrimination is crucial to creating a more just and equitable society, especially over the internet.

The shared task **E**xplainable **D**etection of **O**nline **S**exism *(EDOS)* was part of SemEval 2023 and there were three subtasks associated with it (Kirk et al., 2023). This paper describes the DCU system submitted for all the three subtasks of the EDOS shared task.

Subtask A The first subtask is a straightforward binary classification problem, in which the system has to classify a textual phrase or a sentence as *sexist* or *not sexist*.

Subtask B The second subtask is a mutli-class classification problem, in which the system has to classify a social media post in one of four different categories: *(a)* threats, *(b)* derogation, *(c)* animosity and *(d)* prejudiced discussions.

Subtask C The third subtask is a fine-grained classification problem, in which the system has to classify the sexist textual phrase or sentence in eleven sub-types of the earlier mentioned four categories of Subtask B.

Sections 3 and 4 describe our system in detail. The results of our system are discussed in Section 5. Furthermore, since the shared-task pertains to "explainable" detection of online sexism, we have made an extra effort contributing towards the interpretability of classification systems. To achieve this objective, we recruited two gender studies and one online harm research scholars to annotate ten randomly selected phrases that our submitted system failed to accurately classify as "sexist" and "not sexist". These contributions are discussed in Section 6.

2 Background

We build on fine-tuning pre-trained language models (LMs), a method that has improved many tasks in natural language processing (NLP) over the past four years (Howard and Ruder, 2018; Radford et al., 2018; Devlin et al., 2019; Ruder et al., 2019; Zhang et al., 2021). This method can be categorised as transfer learning as it transfers from one or more pre-training tasks to the target task(s). For transfer learning to be successful, the characteristics of the data used in the pre-training stage should match those of the target task to ensure that the model can transfer relevant knowledge to the new target task. Howard and Ruder (2018) and Gururangan et al. (2020) find that the closer the statistical properties of the pre-training tasks are to the target task, the more likely is success of transfer learning. Furthermore, it has been observed that continued pre-training on in-domain data improves downstream task performance as compared to relying on out-of-domain data only (Gururangan et al., 2020; Sun et al., 2019; Zhang et al., 2021; Karan and Šnajder, 2018; Talat et al., 2018).

Several recent studies have explored the detection of sexist content in social media. Chiril et al. (2020) introduce a unique method for characterising sexist content using speech acts theory. Having collected and annotated 12,000 French tweets, they conduct experiments in sexism detection using convolutional neural networks (CNN), bidirectional recurrent LSTM networks with an attention mechanism and a BERT language model. In contrast, Samory et al. (2021) aim to improve construct validity and reliability in sexism detection by generating adversarial examples. They conduct experiments for sexism detection with CNN and BERT language model and detect both online benevolence and hostility, comparing against Jigsaw's Perspective API (Jigsaw and Google, 2017) as a baseline. Also, Mina et al. (2021) present their sexism identification system for the sexism identification in social networks (EXIST) shared task at IberLEF 2021 (Rodríguez-Sánchez et al., 2021), and they leverage both pre-training and fine-tuning strategies with encoder-only language models, earning them 5th and 6th rank in the shared task.

3 System Overview

Our system is based on two foundational encoder-only pre-trained LMs, and one decoder-only pre-trained LM, described in Section 3.1. Figure 1 shows the components of our system.

The first block shows continued pre-training of the two encoder-only pre-trained LMs with additional in-domain data and is described in Sections 3.2 and 3.3 below. The second block shows fine-tuning for Subtask A, a binary classification task. We fine-tune *(a)* models from the first block and *(b)* off-the-shelf models, both encoder and decoder LMs. The third block depicts a one-vs-all multiclass classification scheme built by fine-tuning an LM on individual categories of Subtask B. Predictions are made based on which label receives the highest probabilities from the four binary classifiers. We experiment with both encoder and decoder LMs. The final block of the system overview shows a fusion of pre-trained LM-based and traditional machine learning methods for Subtask C. This involves training traditional machine learning algorithms, namely logistic regression and nearest neighbor, by aggregating sentence embeddings from both the encoder- and decoder-only pre-trained LMs.

Section 3.4 below discusses the task-specific blocks in more detail. We make our implementation available on `https://github.com/kanishk-adapt/semeval-task10`.

3.1 Foundation Models

As the starting point for continued pre-training and/or fine-tuning, we leverage two encoder-only language models, BERT$_{base-uncased}$ (Devlin et al., 2019) and HateBERT$_{base-uncased}$ (Caselli et al., 2021). In addition, we fine-tune two decoder-only language models, namely Open Pretrained Transformers (OPT) with 350 million and 1.3 billion parameters (Zhang et al., 2022).[1]

3.2 In-domain Datasets

For continued pre-training on in-domain data, we leveraged the combined two million sentences from the unlabelled datasets made available by Kirk et al. (2023). In addition, we leverage approximately one million sentences related to hate speech, online abuse and offensive language from various sources[2] (Davidson et al., 2017; Founta et al., 2018; Kirk et al., 2023; Kolhatkar et al., 2020; Kurrek et al., 2020; Mollas et al., 2022; Mathew et al., 2021; Pavlopoulos et al., 2020; Samory et al., 2021; Waseem, 2016; Wulczyn et al., 2017; Zampieri et al., 2019).

Table 1 shows the size and type of each data set used for continued pre-training.

3.3 Continued Pre-training

For continued pre-training of BERT$_{base-uncased}$, and HateBERT$_{base-uncased}$, we experiment with *(a)* training for both the next sentence prediction task[3] (NSP) and the masked language modelling tasks[4] (MLM), and *(b)* training only with the MLM

[1]We access these models using the Huggingface transformer library (Wolf et al., 2020).

[2]`https://hatespeechdata.com`

[3]Next sentence prediction task is a technique used while training LMs where two sequences are combined as input. The sequences may or may not be next to each other in the original text. Then, the language model has to predict if the two sequences were one after the other or not.

[4]Masked language modelling task is a technique wherein at the time of pretraining 15% of the words in a sentence are

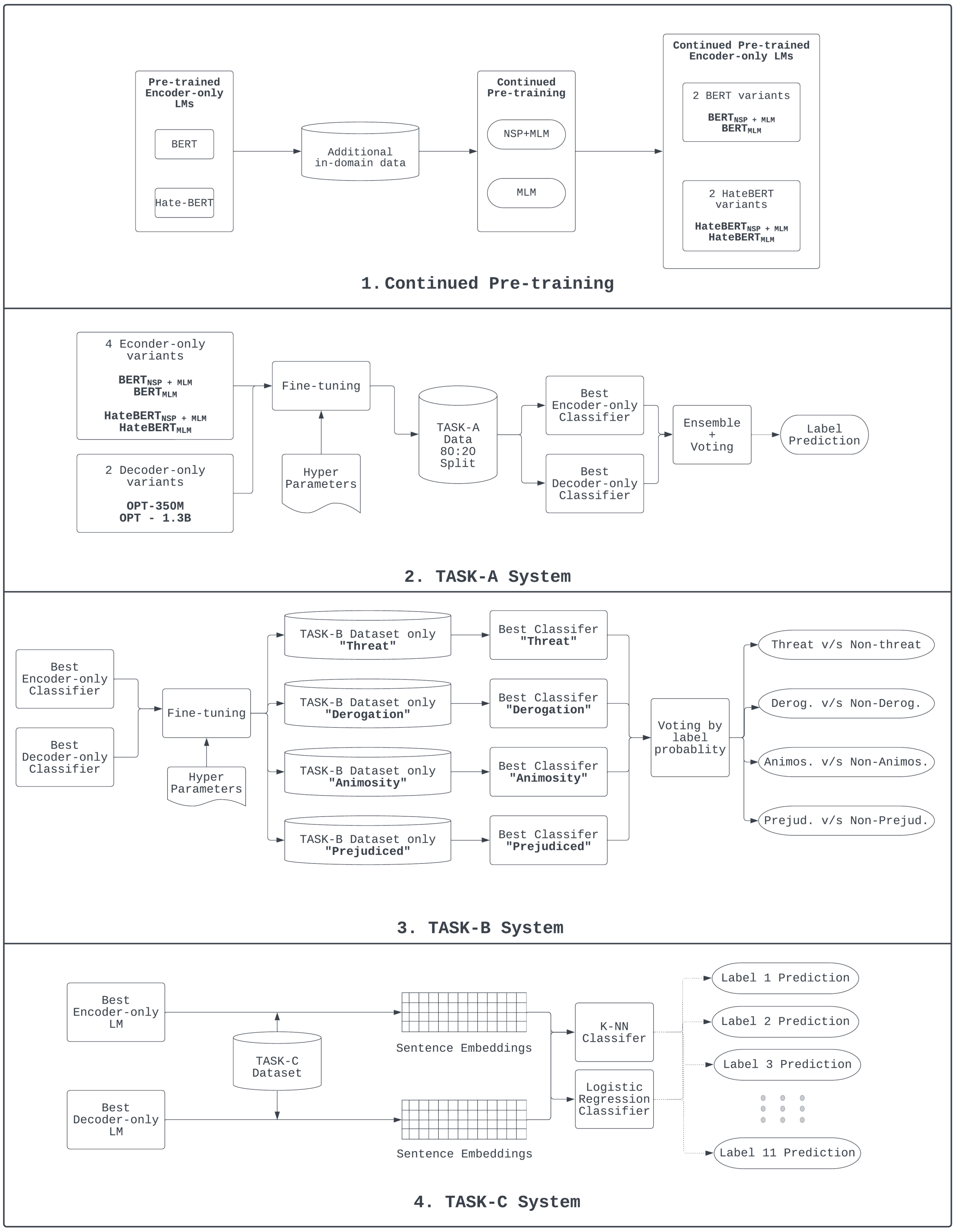

Figure 1: System overview

Author(s)	Size	Dataset Type
Wulczyn et al. (2017)	115,864	Pers. Attacks
Kolhatkar et al. (2020)	692,448	Unh. conv.
Mollas et al. (2022)	998	Hate Speech
Mathew et al. (2021)	20,148	Hate Speech
Zampieri et al. (2019)	13,240	Offensive lang.
Kurrek et al. (2020)	40,003	Abusive lang.
Samory et al. (2021)	13,631	Sexist
Davidson et al. (2017)	24,783	Hate Speech
Waseem (2016)	2,596	Hate Speech
Pavlopoulos et al. (2020)	19,915	Toxic lang.
Founta et al. (2018)	14,871	Abusive lang.
Kirk et al. (2023) GAB	1,000,000	Unlab. dataset
Kirk et al. (2023) Reddit	1,000,000	Unlab. dataset
Total	2,958,497	–

Table 1: Unlabelled data used for continued pre-training and fine-tuning. The size represents number of sentences or posts that were successfully retrieved from the dataset. Pers. = Personal, Unh. conv. = unhealthy conversations, lang. = language, Unlab. = Unlabelled (shared task).

task. The aim of continued pre-training is domain adaptation, i. e. to enhance the ability of the LMs to produce useful representations for internet language, which may include offensive, abusive, or hurtful content. The ultimate objective, however, is to develop a system for the downstream tasks. To select the best pre-trained LMs for fine-tuning, we use perplexity[5] on in-domain data and, for the best epoch only, macro-average F1-score in Subtask A. To measure Subtask A performance, we fine-tune each candidate model by adding a fully connected layer at the top of the candidate model, see details of Subtask A below.

3.4 Target Task LM Fine-tuning

In Subtasks A, B and C of the EDOS challenge, we are asked to perform binary, multi-class and fine-grained classification, respectively. For Subtasks A and B, we adopted the standard approach of fine-tuning both encoder-only and decoder-only models for text classification by placing a fully connected layer over the `[CLS]` output of the LMs. This approach has been widely used for detecting offensive, abusive, or bullying language using encoder-only language models such as BERT, ALBERT, or RoBERTa (Wiedemann et al., 2020), and has produced satisfactory results in English (Zampieri et al., 2020; Ozler et al., 2020; Koufakou et al., 2020; Elsafoury et al., 2021; Verma et al., 2022a,b).

In addition, we fine-tuned the language models for six encoder-only LMs, namely the base versions of BERT$_{base-uncased}$ and HateBERT$_{base-uncased}$, our continued pre-trained encoder-only LMs BERT$_{nsp+mlm}$, BERT$_{mlm+}$, HateBERT$_{nsp+mlm}$ and HateBERT$_{mlm+}$, and two decoder-only LMs, namely OPT-350M and OPT-1.3B.

Subtask A: Binary Classification To classify a given text input as either "sexist" or "not sexist", we fine-tune all versions of the earlier mentioned encoder-only LMs, i. e. the base versions and both the continued pre-trained versions, first using an 80:20 split of the shared task training data. We do so by fine-tuning the pre-trained language models (PLMs) on the target task with a classification head on top of the last output for the `[CLS]` token (as opposed to freezing the PLM parameters and only training the classificaiotn head). We identify the best performing encoder-only language models according to macro-average F1-score. Furthermore, we also fine-tune two decoder-only LMs, OPT-350m and OPT-1.3B on the same split of training data and identify the best-performing one. To determine the range of hyper-parameters to explore in our hyper-parameter search, we examine similar experiments conducted on hate, abusive, bullying and offensive language datasets (Ozler et al., 2020; Koufakou et al., 2020; Elsafoury et al., 2021; Verma et al., 2022a,b). For tracking experiments and selecting the optimal combination of hyper-parameters, we use Weights & Biases (Biewald, 2020) and conduct a grid-search, iterating over all possible combinations of hyper-parameters.

Subtask B: Coarse Classification In Subtask B, the system must classify an input into one of four categories: threat, derogation, animosity, or prejudiced discussion. To achieve this, we use a cascading and voting approach and restrict experiments to the best performing encoder-only and decoder-only LMs from Subtask A according to development set results. The first step of this approach involves transforming the original multi-class classification problem into multiple binary one vs. all classification problems. This is accomplished by creating four distinct sets of training and validation data, with each label being converted into a positive sample while the other labels are treated as negative samples (one versus three classification). For the "threat" model, for example, all instances with the

randomly masked and the language model has to predict the masked words.

[5]Perplexity measures a model's surprise about the next word in a sequence based on the previous words.

label "threat" are positive samples while those with the labels "derogation," "animosity" and "prejudiced discussion" are negative samples. Next, both the encoder-only and decoder-only are fine-tuned for each binary classification task. To determine the best-performing model for each label, the performance of each model is then evaluated for each label on the respective validation set. This process results in a reduction of the number of models used from eight to four. Finally, the best four models are utilized to assign a probability to each label, and the label with the highest probability is selected as the final prediction. This approach enables us to leverage the most accurate model for each label.

Subtask C: Fine-grained Classification This subtask increases the number of categories to be predicted for a given text input to eleven, making this subtask the most complex and nuanced in this shared task. We experiment with logistic regression (LR) and k-nearest neighbor (k-NN) using document vectors as features and two types of document vectors. For the first type of document vectors, we pool the contextual word embeddings of our best fine-tuned encoder-only LM for Subtask A. The second type of document vectors is obtained in the same way but for the the best fine-tuned decoder-only LM for Subtask A. We then train LR and k-NN for each type of document vectors, producing four predictors. We select the best predictor according to development data.

4 Experimental Setup

4.1 Internal Training and Development Sets

Since the labels of the official development set had been withheld for the main part of the development phase, we split the official training into five folds with equal number of training instances and create five internal training sets as for cross-validation. However, due to time constraints and the sufficiently high number of test items in each fold, we only use the first 80:20 split during development, labelled "80:20" below.[6] Cross-validation results labelled "5-fold" below have been obtained after the competition.

4.2 Pre-processing of In-domain Data

In order to further pre-train encoder-only language models, we utilize raw textual data as detailed in Section 3.2. As usernames and URLs were concealed with the placeholders `[user]` and `[url]`, respectively, in the original task dataset, we adopted the same masking technique when preparing in-domain data for pre-training. Moreover, we eliminated non-ASCII characters[7] and retweet markers in Twitter-derived text datasets.

4.3 Training Details

Continued Pre-training Continued pre-training (Section 3.3) is performed on a single Nvidia RTXA6000 GPU. The experiment is tracked using Weights & Biases (Biewald, 2020). Based on previous exploration of hyper-parameters (Devlin et al., 2019; Caselli et al., 2021; Liu et al., 2019), we set the learning rate to 5×10^{-5}, weight decay rate to 0.01, the seed value to ten and batch size to 16, and we train for ten epochs.

Fine-tuning for Subtask A We fine-tune two pre-trained versions each of the two encoder-only LMs (BERT$_{nsp+mlm}$, BERT$_{mlm}$, and HateBERT$_{nsp+mlm}$ and HateBERT$_{mlm}$) on 80% of the training data. The range of hyper-parameters and hardware used for training is described in Appendix A.

The F1-macro scores are observed on the validation set to identify the top-performing LM among the four pre-trained LMs. After shortlisting the best encoder-only pre-trained LM, using the same set of hyper-parameters we fine-tune two decoder-only LMs, OPT-1.3B and OPT-350m. This fine-tuning is conducted on five different subsets of the training data that are created in the manner of a five-fold cross-validation.[8]

Voting in Subtask A After fine-tuning both the best decoder-only and best encoder-only language models (LMs) using five training sets, we select the top five LMs out of the ten LM combinations i. e. five combinations each for best encoder-only and decoder-only LM. To generate the final predictions, we employ a majority voting method by combining the (hard) predictions from these five classifiers. In case of a tie in voting, we randomly pick from the set of labels in the tie.

[6]We use a sequential split with no randomisation as an inspection of the label distribution suggests that the shared task organisers shuffled the data.

[7]Future work should test whether this helps to reduce noise or helpful information is removed, e. g. from emoticons.

[8]We do not evaluate on the held-out data but directly on the shared-task provided development data, except for the first fold for which we evaluate on both development data and held-out data.

Fine-tuning for Subtask B As explained in Section 3.4, we address Subtask B by training four binary classifiers, one for each label, and training in the context of PLMs means fine-tuning. We fine-tune both the best-performing encoder-only and the best-performing decoder-only LMs for the four binary classification tasks. Subsequently, we assess the performance of each fine-tuned model using F1-macro in the binary classification task it was tuned for. We then pick, for each of the four target labels, the best model from the available two models that have been tuned for the binary classification task for that target label. The selected four models are then used as described in Section 3.4 to make the final prediction.

Training for Subtask C For Subtask C, we train k-NN and LR classifiers on document embeddings (Section 3.4) for each of our five subsets of 80% of the training data created in the manner of 5-fold cross-validation. For LR, we tune the penalty term of L2 regularization, also known as the ridge classifier, with a grid-search and cross-validation.

5 Results

In this section, we present results from our development steps of building our system.[9] All tables show F1-macro average scores in percentages.

5.1 Continued Pre-training Results

Our results for continued pre-training on in-domain are depicted in Figures 2 and 3. For continued pre-training with both NSP and MLM tasks, we observe a reduction of LM performance in terms of both LM perplexity (Salazar et al., 2020) and efficiency in downstream tasks, here Subtask A, in Figure 3. On the contrary, continued pre-training with only the MLM task improves the performance by more than 2% F1-macro score compared to the base models (Figure 3).

The left-most sub-figures of Figure 2 show that continued pre-training with only the MLM objective reduces perplexity, whereas right-most sub-figures in the same figure show an increase in perplexity with epoch progression for the standard BERT pre-training objective combining MLM and NSP. This indicates that the combined NSP and MLM training objective is not suitable for the selected data. We speculate that this is due to the dominance of very short documents in the data not providing the needed continuity for the NSP task.

[9]Appendix B provides baselines using hand-crafted features as a control for errors in training our PLM-based systems.

Type	Model	Type	F1
Encoder-only	BERT	base-uncased	81.39
		NSP+MLM	80.09
		MLM	82.89
Encoder-only	HateBERT	base-uncased	82.52
		NSP+MLM	81.45
		MLM	**84.30**
Decoder-only	OPT	350 M	82.61
		1.3 B	**84.10**

Table 2: Subtask A results on 80:20 split. NSP = Next Sentence Prediction, MLM = Masked Language Modelling.

Model	Type	Dev. F1	Test F1
HateBERT	MLM	84.30	-
OPT	1.3 B	84.83	-
HateBERT + $\text{OPT}_{1.3B}$	Voting	**84.91**	**85.59**

Table 3: Subtask A development and test set results. The HateBERT LM is HateBERT_{mlm+}.

Overall, the results shown in Figures 2 and 3 indicate that HateBERT_{mlm+} is the best performing encoder-only LMs in our experiment.

5.2 Results for Target Tasks

Subtask A As previously stated in Sections 3.3 and 3.4, we utilize the F1-macro score as a metric to assess the impact of domain adaptation with continued pre-training on our classifier's performance. Table 2 shows our results obtained on the test set of our "80:20" split, thereby revealing the best-performing LMs for the task among our choice of encoder-only and decoder-only LMs. We see that all the models are competitive with comparable F1-macro scores. However, HateBERT_{mlm+} and OPT-1.3B appears to give the overall best performance of the encoder-only and decoder-only LMs respectively. In Table 7 (Appendix C), we provide the results of both models under 5-fold cross-validation. In Table 3, we observe that both models achieve similar F1 scores on the development set task labels. By applying the voting mechanism discussed in Section 3.4, we achieve a performance improvement of nearly 1% on the development set and an F1 score of **0.8559** on the test set which places us 16th among the 89 participating teams for Subtask A.

Subtask B Our strategy for Subtask B involves assessing the performance of HateBERT_{mlm+} and OPT-1.3B for each label on our "80:20" split of training data, and the results are presented in Ta-

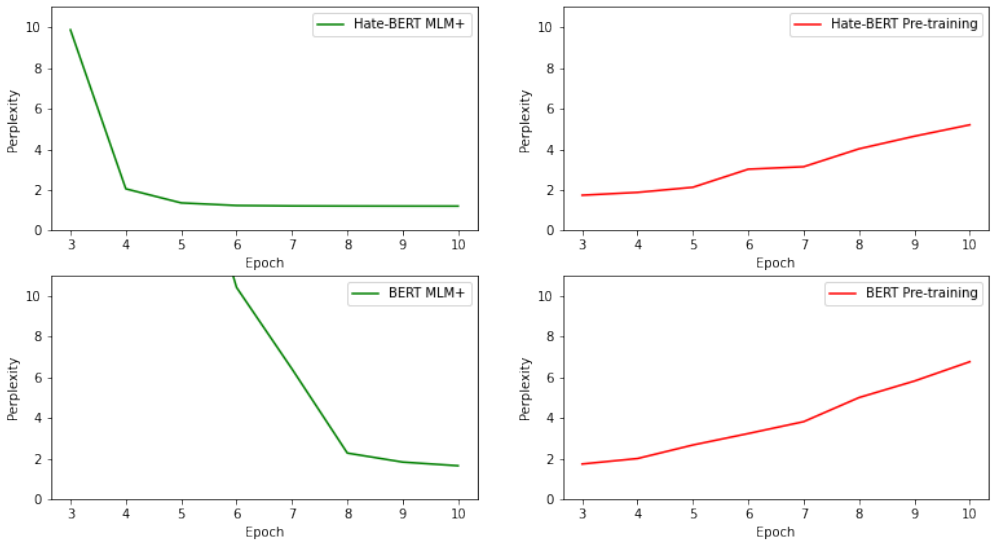

Figure 2: Perplexity scores of each continued pre-training strategy (from 3rd epoch onwards and y-axes limited to (0,10) for comparability)

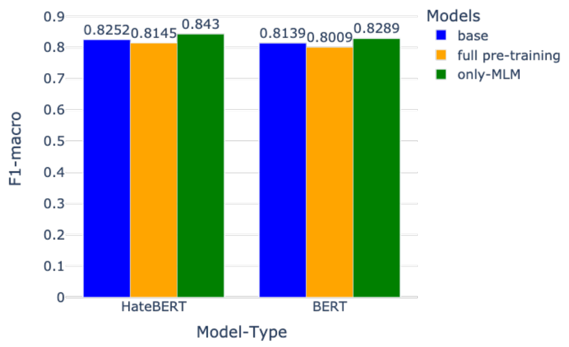

Figure 3: Performance of LMs with both continued pre-training and fine-tuning of base models based on F1-macro score grouped by LMs

ble 4. We observe that HateBERT$_{mlm+}$ outperforms OPT-1.3B for each label, indicating that encoder-only language models are better suited for our one-vs-all approach than decoder-only LMs, such as OPT-1.3B. To determine which combinations of language models would improve our one-vs-all approach for Subtask B, we conducted 5-fold cross-validation. The results, shown in Table 5, indicate that our best-performing combination is with HateBERT$_{mlm+}$, resulting in an F1-score of 0.54 on the development set and 0.5104 on the test set. Our approach earned us a ranking of 59th out of the 69 participating teams for Subtask B.

Label	**HB**	**OPT**
Threats.	**85.09**	48.14
Derog.	**66.77**	35.08
Animosity	**65.73**	38.66
Prejudice.	**71.48**	47.40

Table 4: F1 scores of the binary classifiers considered as candidates for the Subtask B system, trained and tested using our primary 80:20 split. HB = HateBERT$_{mlm+}$, OPT = OPT-1.3B, Threats = 1. Threats, plans to harm and incitement, Derog. = 2. Derogation, Animosity = 3. Animosity, Prejudice. = 4. Prejudiced Discussions.

Model	**Dev. F1**	**Test F1**
HateBERT$_{+}$	**54.00**	**51.04**
OPT-1.3B	24.00	-

Table 5: Subtask B development and test set macro F1 scores for the combination of HateBERT-based classifiers and for the corresponding OPT-1.3B-based classifiers.

LM	**Trad. ML**	**Dev. F1**	**Test F1**
OPT-1.3B	LR	15.35	-
	K-NN	15.00	-
HateBERT$_{mlm+}$	**LR**	**43.16**	**37.95**
	K-NN	40.16	-

Table 6: Subtask C results on Development and Test-set {LM. = Language Model, Trad. ML = Traditional Machine Learning technique, Dev. F1 = Development set F1-macro, Test. F1 = Test set F1-macro, LR. = Logisitic Regression, K-NN = K-Nearest Neighbour}

Subtask C Our results in Table 6 show good performance for the encoder-only LM HateBERT$_{mlm+}$ with both machine learning methods, yielding F1-macro scores over 40%. The best performance is reached with LR, three percentage points above k-NN. With our decoder-only LM, however, the F1-macro score is considerably lower, just over 15%. On the test set, our simple fusion approach, which involved using HateBERT$_{mlm+}$ sentence embeddings and Logistic Regression, yields an F1-macro score of 0.3795, earning our team a ranking of 38th out of 63 participating teams. These results indicate the effectiveness of our strategy for Subtask C and suggest the potential for further exploration of encoder-only LMs in other natural language processing tasks.

6 Towards "Explainable" Online Sexism Detection

Taking steps towards "explainable" detection of sexism, we analysed the decisions made by our submitted system for Subtask A, where it incorrectly classified certain phrases or sentences as either "sexist" or "not-sexist" as annotated by (Kirk et al., 2023). To achieve this, we randomly select ten samples from the test set that our submitted system failed to classify accurately. Next, we calculate the mean gradient importance scores for our best-performing encoder-only LM, HateBERT (Verma et al., 2022b; Elsafoury et al., 2021). These scores are calculated using the Integrated Gradients algorithm (Sundararajan et al., 2017) for PyTorch (Kokhlikyan et al., 2020). To provide a zero-shot comparison with our system, we also use Flan-T5$_{large}$ (Chung et al., 2022) and ChatGPT.[10] In order to extract signals or rationales for the prediction, we use input prompts as discussed in Appendix D. Taking an adversarial approach, we hypothesise that there is a bit of ambiguity in the ground truth label released by the competition organizer.

In order to validate our hypothesis and provide an explainable analysis, we recruit three social scientists with expertise in gender studies to manually assign labels to the selected samples. The task organizers employed the descriptive approach (Rottger et al., 2022) for their annotation, but at the time of our experiment, the annotation guidelines were not publicly available. Consequently, we utilized the definition of sexist content provided in the task description to create our own annotation guidelines outlined in Appendix E. Our annotation guidelines provide clear instructions for annotators to follow, instructing them to, *(a)* mark if the text is `sexist` or `not sexist`, *(b)* select words that led to their decision, and *(c)* rate their level of agreement with the generated rationale by Flan-T5 and ChatGPT.

Our findings reveal that the inter-rater reliability score among the annotators was 0.4821 for *(a)*, calculated using Krippendorff alpha. Furthermore, the annotators disagreed with the original annotations in **four** out of the **five** phrases that were originally annotated as "not-sexist". This suggests that the ground truth annotations may not have been entirely accurate and that there is a need for more nuanced and detailed annotations for phrases marked as "not-sexist". Considering how subjective and emotive a gender-based subject could be, we recommend that future dataset should be annotated or at least validated by persons with considerable expertise in gender-based research.

After the competition, we were able to investigate our hypothesis with the dataset released by Kirk et al. (2023). Our analysis revealed that there is ambiguity in the ground truth labels provided by the competition organisers. For instance, the sentence - id "`sexism2022_english-15683`", in the test split was individually annotated as "`sexist`" by all three annotators recruited by the task organisers. Our expert annotators also agreed with this assessment. However, the aggregated labels shared by the task organisers classified the sentence as "`not sexist`". Although (Kirk et al., 2023) state that expert adjudication is reserved for cases with less than 3/3 agreement (unanimous) in Subtask A, our findings indicate that there is still some degree of ambiguity in the ground truth labels provided by the competition organisers.

7 Conclusion

We experimented with two encoder-only and one decoder-only language models (LMs) for the shared-task, including BERT (Devlin et al., 2019), HateBERT (Caselli et al., 2021), and OPT (Zhang et al., 2022). We also explored continued pre-training strategies for encoder-only LMs, namely BERT and HateBERT. Our observations support the finding by Liu et al. (2019) that removing the next sentence prediction task during pre-training improves the LM's performance. Our results indi-

[10]A chat product based on the work of Ouyang et al. (2022) made available by OpenAI `https://openai.com/blog/chatgpt`

cate that both decoder-only and encoder-only LMs, namely HateBERT$_{mlm+}$ and OPT-1.3B, yield similar results when fine-tuned by placing a fully connected layer over the `[CLS]` output for binary classification. Moreover, we improved Subtask A performance with a simple voting mechanism, building an ensemble of both the encoder- and decoder-based classifiers. In Subtask B, we find a surprisingly large performance difference between fine-tuning HateBERT and fine-tuning OPT in our basic one-vs-all approach, with OPT having difficulties in all four binary classification sub-problems. Additionally, we found that our approach of fusing traditional machine learning algorithms for classification with document representations obtained with encoder-only LMs works well for Subtask C. Although this approach is simple, our rank indicates that it can be further improved upon.

Furthermore, our study highlights the importance of interpretability and explainability in detection systems, particularly for sensitive issues such as sexism. By using a combination of methods such as input prompts, annotator feedback, and gradient importance scores, we may gain a better understanding of how these systems make decisions and identify areas for improvement. This can ultimately lead to more reliable and accurate detection of sexism and other forms of discrimination.

Acknowledgements

We acknowledge and express our sincere gratitude to the domain expert annotators who generously dedicated their time and expertise to annotate the sentences on short notice. The research conducted in this publication was funded by the Irish Research Council and Google, Ireland, under grant number EPSPG/2021/161. The second author has received funding from the European Union's Horizon 2020 research and innovation programme under the Marie Sklodowska-Curie grant agreement No. 847402. This research was also conducted with the financial support of Science Foundation Ireland under Grant Agreement No. 13/RC/2106_P2 at the ADAPT SFI Research Centre at Dublin City University. ADAPT, the SFI Research Centre for AI-Driven Digital Content Technology, is funded by Science Foundation Ireland through the SFI Research Centres Programme. For the purpose of Open Access, the author has applied a CC BY public copyright licence to any Author Accepted Manuscript version arising from this submission.

References

Maja Brandt Andreasen. 2021. 'rapeable' and 'unrapeable' women: the portrayal of sexual violence in internet memes about# metoo. *Journal of Gender Studies*, 30(1):102–113.

Lukas Biewald. 2020. Experiment tracking with weights and biases. Software available from wandb.com.

Steven Bird and Edward Loper. 2004. NLTK: The natural language toolkit. In *Proceedings of the ACL Interactive Poster and Demonstration Sessions*, pages 214–217, Barcelona, Spain. Association for Computational Linguistics.

Steven Bird, Edward Loper, and Ewan Klein. 2009. *Natural Language Processing with Python*. O'Reilly Media Inc.

Tommaso Caselli, Valerio Basile, Jelena Mitrović, and Michael Granitzer. 2021. HateBERT: Retraining BERT for abusive language detection in English. In *Proceedings of the 5th Workshop on Online Abuse and Harms (WOAH 2021)*, pages 17–25, Online. Association for Computational Linguistics.

Patricia Chiril, Véronique Moriceau, Farah Benamara, Alda Mari, Gloria Origgi, and Marlène Coulomb-Gully. 2020. An annotated corpus for sexism detection in French tweets. In *Proceedings of the Twelfth Language Resources and Evaluation Conference*, pages 1397–1403, Marseille, France. European Language Resources Association.

Hyung Won Chung, Le Hou, Shayne Longpre, Barret Zoph, Yi Tay, William Fedus, Eric Li, Xuezhi Wang, Mostafa Dehghani, Siddhartha Brahma, et al. 2022. Scaling instruction-finetuned language models. ArXiv 2210.11416v5.

Thomas Davidson, Dana Warmsley, Michael Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language. *Proceedings of the International AAAI Conference on Web and Social Media*, 11(1):512–515.

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.

Jessica Drakett, Bridgette Rickett, Katy Day, and Kate Milnes. 2018. Old jokes, new media–online sexism and constructions of gender in internet memes. *Feminism & psychology*, 28(1):109–127.

Fatma Elsafoury, Stamos Katsigiannis, Zeeshan Pervez, and Naeem Ramzan. 2021. When the timeline meets the pipeline: A survey on automated cyberbullying detection. *IEEE access*, 9:103541–103563.

Antigoni-Maria Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Sirivianos, and Nicolas Kourtellis. 2018. Large scale crowdsourcing and characterization of Twitter abusive behavior. *Proceedings of the International AAAI Conference on Web and Social Media*, 12(1).

Suchin Gururangan, Ana Marasović, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don't stop pretraining: Adapt language models to domains and tasks. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360, Online. Association for Computational Linguistics.

Matthew Honnibal and Mark Johnson. 2015. An improved non-monotonic transition system for dependency parsing. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1373–1378, Lisbon, Portugal. Association for Computational Linguistics.

Jeremy Howard and Sebastian Ruder. 2018. Universal language model fine-tuning for text classification. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 328–339, Melbourne, Australia. Association for Computational Linguistics.

Clayton Hutto and Eric Gilbert. 2014. Vader: A parsimonious rule-based model for sentiment analysis of social media text. *Proceedings of the International AAAI Conference on Web and Social Media (ICWSM)*, 8(1):216–225.

Jigsaw and Google. 2017. Perspective API.

Mladen Karan and Jan Šnajder. 2018. Cross-domain detection of abusive language online. In *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, pages 132–137, Brussels, Belgium. Association for Computational Linguistics.

Hannah Rose Kirk, Wenjie Yin, Bertie Vidgen, and Paul Röttger. 2023. SemEval-2023 Task 10: Explainable detection of online sexism. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, Toronto, Canada. Association for Computational Linguistics.

Narine Kokhlikyan, Vivek Miglani, Miguel Martin, Edward Wang, Bilal Alsallakh, Jonathan Reynolds, Alexander Melnikov, Natalia Kliushkina, Carlos Araya, Siqi Yan, and Orion Reblitz-Richardson. 2020. Captum: A unified and generic model interpretability library for pytorch. ArXiv 2009.07896v1.

Varada Kolhatkar, Hanhan Wu, Luca Cavasso, Emilie Francis, Kavan Shukla, and Maite Taboada. 2020. The SFU opinion and comments corpus: A corpus for the analysis of online news comments. *Corpus Pragmatics*, 4:155–190.

Anna Koufakou, Endang Wahyu Pamungkas, Valerio Basile, and Viviana Patti. 2020. HurtBERT: Incorporating lexical features with BERT for the detection of abusive language. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 34–43, Online. Association for Computational Linguistics.

Jana Kurrek, Haji Mohammad Saleem, and Derek Ruths. 2020. Towards a comprehensive taxonomy and large-scale annotated corpus for online slur usage. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 138–149, Online. Association for Computational Linguistics.

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 1907.11692v1.

Kate Manne. 2017. *Down girl: The logic of misogyny*. Oxford University Press.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2021. Hatexplain: A benchmark dataset for explainable hate speech detection. *Proceedings of the AAAI Conference on Artificial Intelligence*, 35(17):14867–14875. IAAI-21, EAAI-21, AAAI-21 Special Programs and Special Track.

Schütz Mina, Boeck Jaqueline, Liakhovets Daria, Slijepčević Djordje, Kirchknopf Armin, Hecht Manuel, Bogensperger Johannes, Schlarb Sven, Schindler Alexander, and Zeppelzauer Matthias. 2021. Automatic sexism detection with multilingual transformer models: AIT FHSTP@EXIST2021. In *Proceedings of the Iberian Languages Evaluation Forum (IberLEF 2021), co-located with the Conference of the Spanish Society for Natural Language Processing (SEPLN 2021), XXXVII International Conference of the Spanish Society for Natural Language Processing*, pages 346–355, Málaga, Spain.

Ioannis Mollas, Zoe Chrysopoulou, Stamatis Karlos, and Grigorios Tsoumakas. 2022. ETHOS: a multi-label hate speech detection dataset. *Complex & Intelligent Systems*.

Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, John Schulman, Jacob Hilton, Fraser Kelton, Luke Miller, Maddie Simens, Amanda Askell, Peter Welinder, Paul F Christiano, Jan Leike, and Ryan Lowe. 2022. Training language models to follow instructions with human feedback. In *Advances in Neural Information Processing Systems*, volume 35, pages 27730–27744. Curran Associates, Inc.

Kadir Bulut Ozler, Kate Kenski, Steve Rains, Yotam Shmargad, Kevin Coe, and Steven Bethard. 2020. Fine-tuning for multi-domain and multi-label uncivil language detection. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 28–33, Online. Association for Computational Linguistics.

John Pavlopoulos, Jeffrey Sorensen, Lucas Dixon, Nithum Thain, and Ion Androutsopoulos. 2020. Toxicity detection: Does context really matter? In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 4296–4305, Online. Association for Computational Linguistics.

Alec Radford, Karthik Narasimhan, Tim Salimans, and Ilya Sutskever. 2018. Improving language understanding by generative pre-training. OpenAI Preprint.

Radim Řehůřek and Petr Sojka. 2010. Software framework for topic modelling with large corpora. In *Proceedings of the LREC 2010 Workshop on New Challenges for NLP Frameworks*, pages 46–50, Valletta, Malta. ELRA.

Francisco Rodríguez-Sánchez, Jorge Carrillo-de Albornoz, Laura Plaza, Julio Gonzalo, Paolo Rosso, Miriam Comet, and Trinidad Donoso. 2021. Overview of EXIST 2021: sexism identification in social networks. *Procesamiento del Lenguaje Natural*, 67:195–207.

Paul Rottger, Bertie Vidgen, Dirk Hovy, and Janet Pierrehumbert. 2022. Two contrasting data annotation paradigms for subjective NLP tasks. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 175–190, Seattle, United States. Association for Computational Linguistics.

Sebastian Ruder, Matthew E. Peters, Swabha Swayamdipta, and Thomas Wolf. 2019. Transfer learning in natural language processing. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Tutorials*, pages 15–18, Minneapolis, Minnesota. Association for Computational Linguistics.

Julian Salazar, Davis Liang, Toan Q. Nguyen, and Katrin Kirchhoff. 2020. Masked language model scoring. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 2699–2712, Online. Association for Computational Linguistics.

Mattia Samory, Indira Sen, Julian Kohne, Fabian Flöck, and Claudia Wagner. 2021. "call me sexist, but...": Revisiting sexism detection using psychological scales and adversarial samples. *Proceedings of the International AAAI Conference on Web and Social Media*, 15(1):573–584.

Chi Sun, Xipeng Qiu, Yige Xu, and Xuanjing Huang. 2019. How to fine-tune bert for text classification? In *Chinese Computational Linguistics: 18th China National Conference, CCL 2019, Kunming, China, October 18–20, 2019, Proceedings 18*, pages 194–206. Springer.

Mukund Sundararajan, Ankur Taly, and Qiqi Yan. 2017. Axiomatic attribution for deep networks. In *Proceedings of the 34th International Conference on Machine Learning*, volume 70 of *Proceedings of Machine Learning Research*, pages 3319–3328. PMLR.

Zeerak Talat, James Thorne, and Joachim Bingel. 2018. Bridging the gaps: Multi task learning for domain transfer of hate speech detection. *Online Harassment*, pages 29–55.

Kanishk Verma, Tijana Milosevic, Keith Cortis, and Brian Davis. 2022a. Benchmarking language models for cyberbullying identification and classification from social-media texts. In *Proceedings of the First Workshop on Language Technology and Resources for a Fair, Inclusive, and Safe Society within the 13th Language Resources and Evaluation Conference*, pages 26–31, Marseille, France. European Language Resources Association.

Kanishk Verma, Tijana Milosevic, and Brian Davis. 2022b. Can attention-based transformers explain or interpret cyberbullying detection? In *Proceedings of the Third Workshop on Threat, Aggression and Cyberbullying (TRAC 2022)*, pages 16–29, Gyeongju, Republic of Korea. Association for Computational Linguistics.

Zeerak Waseem. 2016. Are you a racist or am I seeing things? annotator influence on hate speech detection on Twitter. In *Proceedings of the First Workshop on NLP and Computational Social Science*, pages 138–142, Austin, Texas. Association for Computational Linguistics.

Gregor Wiedemann, Seid Muhie Yimam, and Chris Biemann. 2020. UHH-LT at SemEval-2020 task 12: Fine-tuning of pre-trained transformer networks for offensive language detection. In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1638–1644, Barcelona (online). International Committee for Computational Linguistics.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Wikipedia talk labels: Personal attacks. Figshare 4054689.

Marcos Zampieri, Shervin Malmasi, Preslav Nakov, Sara Rosenthal, Noura Farra, and Ritesh Kumar. 2019. Predicting the type and target of offensive posts in social media. 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 1415–1420, Minneapolis, Minnesota. Association for Computational Linguistics.

Marcos Zampieri, Preslav Nakov, Sara Rosenthal, Pepa Atanasova, Georgi Karadzhov, Hamdy Mubarak, Leon Derczynski, Zeses Pitenis, and Çağrı Çöltekin. 2020. SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020). In *Proceedings of the Fourteenth Workshop on Semantic Evaluation*, pages 1425–1447, Barcelona (online). International Committee for Computational Linguistics.

Jing Zhang, Yimeng Zhuang, and Yinpei Su. 2021. TA-MAMC at SemEval-2021 task 4: Task-adaptive pretraining and multi-head attention for abstract meaning reading comprehension. In *Proceedings of the 15th International Workshop on Semantic Evaluation (SemEval-2021)*, pages 51–58, Online. Association for Computational Linguistics.

Susan Zhang, Stephen Roller, Naman Goyal, Mikel Artetxe, Moya Chen, Shuohui Chen, Christopher Dewan, Mona Diab, Xian Li, Xi Victoria Lin, Todor Mihaylov, Myle Ott, Sam Shleifer, Kurt Shuster, Daniel Simig, Punit Singh Koura, Anjali Sridhar, Tianlu Wang, and Luke Zettlemoyer. 2022. Opt: Open pre-trained transformer language models. ArXiv 2205.01068v4.

A Hyper-parameters & Hardware

We explore the following hyper-parameters for fine-tuning:

- Learning rate: $k \times 10^{-5}$ with $k = 1, 2, 3, 4, 5$
- Weight-decay rate: 0.1, 0.01, 0.001
- Batch-size: 8, 16, 32
- Token Maximum Length: 128, 256
- Epochs: 2, 4, 6

We used a single RTXA6000 GPU for fine-tuning both encoder and decoder-only LMs.

B Baseline System

We train baseline systems with Naïve Bayes, Decision Trees, Random Forests, Linear support vector machines and XGBoost using hand-crafted features described below.

Most of our hand-crafted features are n-gram features extracted from the text input. For our feature set, we consider combinations of unigram, bigram and trigram features. We extract both case-sensitive and case-insensitive n-grams features and apply a frequency cut-off to exclude n-grams that are rare in the training data. We try two variants of n-gram features: *(a)* binary indicator features marking the presence of each n-gram and *(b)* integer features counting the number of times each n-gram occurs in the input. For all n-gram features, we considered whether to add $n-1$ padding tokens at the start and end of each input text. Other features include n-grams over POS sequences, dependency labels and token-level sentiment predictions, and the number of input tokens that appear in clusters of *domain words* (one feature per cluster). To build our word clusters, we first select all content words from the training items labelled as sexist with *(a)* negative and *(b)* positive sentiment polarity[11] and their in-context word vectors according to HateBERT and OPT. The word vectors for each polarity and PLM are then partitioned into five clusters of semantically related domain words, producing $2 \times 2 \times 5 = 20$ word lists in total.

We compare results for the following four tokenisers as the choice of text preprocessing influences what features are being extracted: simply splitting at whitespace, NLTK[12] (Bird and Loper, 2004; Bird et al., 2009), spaCy[13] (Honnibal and Johnson, 2015) and Gensim[14] (Řehůřek and Sojka, 2010). Gensim in particular has a big impact on the extracted n-gram features as it deletes all punctuation.

Dependency Parsing To extract syntactic dependencies within text data, we leverage the SpaCy library (Honnibal and Johnson, 2015) and the `en_core_web_trf` statistical model which includes Roberta-base features to extract dependency tags.

Sentiment Prediction Each token in the text is labelled with a sentiment polarity (positive, negative or neutral) using the library NLTK Vader (Hutto and Gilbert, 2014).

Soft Clipping of Frequency Counts We experiment with raising frequency counts to the power of α with $0 < \alpha < 1$ to obtain feature values. For α approaching 0 ($\lim_{\alpha \to 0}$), non-zero counts are mapped close to 1, making the features similar to binary indicator features. For α approaching 1 ($\lim_{\alpha \to 1}$), the identity function is approximated.

[11]Annotated with NLTK Vader (Hutto and Gilbert, 2014)

[12]`https://www.nltk.org/`

[13]`https://spacy.io`

[14]`https://radimrehurek.com/gensim/intro.html`

For $\alpha = 0.5$, the square root of each frequency count is used, reducing the magnitude of feature values.[15]

Data Augmentation We explore data augmentation using the concatenation of three documents with the same fine-grained Subtask C label.[16]

As the concatenation of documents increases the expected magnitude of our hand-crafted feature vectors, we experiment with normalisation of feature vectors before or after clipping frequency counts to binary presence indicators.

Experimental Setup As the purpose of the baseline system is only to provide a lower bound that indicates a problem with a PLM-based system if the PLM-based system's performance falls below the lower bound, we only perform light, unsystematic tuning of hyper-parameters, tuning each hyper-parameter as the corresponding feature is implemented. We start with comparing a few variants of Naïve Bayes, including the use of a uniform prior and a prior set according to the label distribution in the training data. Then we consider decision trees and tune the minimum leaf size, and we extend the approach to random forests. Finally, we add n-gram features based on POS tags, dependency labels and word-level sentiment polarity, and add linear support vector machines and XGBoost as classifiers.

Subtask A Baseline Results Our first baseline macro F1 for Subtask A was 69.20% with Naive-Bayes and `string.split()` as tokeniser, i. e. without separation of punctuation, using our 80:20 split. Using the gensim tokeniser that skips punctuation, F1 improves to 69.50%. Switching to decision trees and tuning the minimum leaf size, F1 further improves to 72.90. Adding POS tag, dependency and sentiment n-gram features, we improve to 73.70. However, we find that using more features is not always better as we get degraded performance of 71.60 when combining tags to complex tags such as "NOUN:positive" before building n-grams. With some more feature tuning, including the use of case-insensitive n-grams, F1 growth to 74.62.

Subtask B Baseline Results Our first Subtask B baseline macro F1 with XGBoost is 44.12%, It is not clear from our logs whether this is for our 80:20 split or on the official development set. Re-evaluating with the best three Subtask A classifiers and different types of data augmentation, we find better performance with LinearSVM, data augmentation based on document concatenation and normalisation of the frequency counts by the number of documents concatenated: macro F1 49.08%. However, with document augmentation, we find that the random seed that influences the random samples of documents to be concatenated affects performance a lot. In 25 repetitions with different random seeds, macro F1 ranges from 46.51% to 50.51% (average 48.37%) trained on 80% of training data but tested on the official development set.

Training on the full official training data, the development macro F1 average over 25 runs drops to 47.32% (range 45.94% to 49.45%). This confirms our observations with PLMs that adding the last 20% of training data is harmful.

Subtask C Baseline Results Our first Subtask C baselines macro F1 with XGBoost is 23.01%. As for Subtask B, it is not clear which development set was used. Our first Subtask C result training on the full training set and testing on the official development data is 23.87%.

Trying to boost this performance with an ensemble of classifiers, we observe a degradation of performance. Analysing the predictions of individual classifiers and the ensemble for cases where performance drops, we find that for each test item only a different minority of classifiers matches the gold label and often the majority is wrong. For example, if there are three test items and classifier i predicts the correct label for item i, individual accuracy will be 1/3 but ensemble accuracy will be zero. We conclude that a simple majority voting ensemble should always be evaluated before use as it cannot be assumed to never harm performance.

With 25 runs, we get an average macro F1 of 23.25% when training on 80% and testing on the official development data (range 22.15% to 24.94%). This mildly improves to 23.58% (range 22.12% to 25.47%) when training on the full training set.

C Cross-validation Results

Table 7 shows cross-validation results in Subtask A.

[15]This transformation has no effect on decision tree learning as the ranking of feature values is not changed.

[16]We also tried using the task B label in task B but this did not perform as well as using the task C label for choosing the candidate documents for concatenation.

Fold	F1-macro-average	
	HateBERT$_{mlm+}$	OPT-1.3B
1	83.46	84.83
2	82.45	**84.64**
3	82.91	84.29
4	**84.30**	84.07
5	83.62	84.44
Average	83.35	84.45

Table 7: Subtask A results on five-fold cross-validation

D Prompts

D.1 Chat-GPT Prompt

Imagine you are an ethics control officer and would like to identify sexist comments and provide a rationale for your prediction. I will be providing you some text, your task will be to analyse if the text is sexist or not sexist. You can only predict one of both categories, either the text is sexist or neutral. Your second task is to identify any words or signals in the text which informs your prediction. The third task is to provide a rationale for your prediction, i.e., why you think the text is sexist or not. You should provide your response in a standard json format. Do you understand the task?

D.2 Flan-T5 Prompt

We use the following prompts for Flan-T5:

- Answer the following question, identify if the following sentence is "sexist" or "not sexist", Sentence: `[input sentence]`

- Answer the following question, identify key phrases or words or signals from the `[sexist or not-sexist]` sentence `[input sentence]`

- Answer the following question, describe why the following sentence is `[sexist or not-sexist]`. Start by writing I base my decision on…

E Annotation Guideline

Step 1

- Start by reading the text, the prediction made by the two systems, lets call them System-A and System-B.

- The prediction is either "sexist" or "not sexist" i.e., whether or not the text has abusive or negative sentiment directed towards women or their gender.

- Each text has a unique "ID" which you can ignore.

- You can find the text in column B, predictions made by System-A and B in columns C and E respectively.

Step 2

- After you have read the sentence, the predictions, take some time to evaluate your level of agreement or disagreement with the prediction of System-A and B.

- You can use a 5-point likert scale to rate your level of agreement, with 1 meaning strongly disagree and 5 meaning strongly agree. For example, if you completely agree with the prediction and rationale, you would rate it as a 5-Strongly Agree. If you're not sure or have some reservations, you could rate it as a 3-Not sure.

- You can use the columns D and F in the spreadsheet to rate your level of agreement with predictions made by System-A and B.

Step 3

- In addition to rating your agreement level, we ask you to a) provide your decision if the text is sexist or not-sexist, and b) identify any words or phrases in the text which informs your decision.

- You can use columns G and H in the spreadsheet to provide your decision and to select words or phrases which inform the decision, respectively.

- For example, the text "women are drama queen", has a negative sentiment directed towards women hence it can be sexist, and the words "drama queen", present in the text inform my agreement. So, I will mark the text as "sexist" in column G and add the words "drama queen" in column H.

- Please note that the task organisers defined "sexist" as "any abuse or negative sentiment that is directed towards women based on their gender, or based on their gender combined with one or more other identity attributes (e.g. Black women, Muslim women, Trans women)". So please refer to this definition while providing your decision.

Step 4

- Once you've completed Step-3, take some time to go through the columns I and K which represent the rationale for the decision making by System-A and System-B.
- You have an option to provide a written explanation of your agreement with the rationale by System-A and System-B in columns I and K respectively. You can start by writing "For instance, I think. . . ."