SKAM at SemEval-2023 Task 10: Linguistic Feature Integration and Continuous Pretraining for Online Sexism Detection and Classification

Murali Manohar Kondragunta **Amber Chen** **Karlo Slot**
Sanne Weering **Tommaso Caselli**
University of Groningen
{m.m.c.kondragunta, a.chen.1,
k.h.r.slot, s.weering}@student.rug.nl
t.caselli@rug.nl

Abstract

Sexism has been prevalent online. In this paper, we explored the effect of explicit linguistic features and continuous pretraining on the performance of pretrained language models in sexism detection. While adding linguistic features did not improve the performance of the model, continuous pretraining did slightly boost the performance of the model in Task B from a mean macro-F1[1] score of 0.6156 to 0.6246. The best mean macro-F1 score in Task A was achieved by a finetuned HateBERT model using regular pretraining (0.8331). We observed that the linguistic features did not improve the model's performance. At the same time, continuous pretraining proved beneficial only for nuanced downstream tasks like Task-B.

1 Introduction

Currently, a lot of models are proposed to flag online content containing sexism. However, these models only flag *what* is sexist content, but not explain *why*. Categorization of sexism will give a more detailed explanation as to why certain expressions are sexism. It leads to a better understanding of the decisions that the model makes, which can improve the model further. To overcome this limitation, SemEval 2023 Task 10 - Explainable Detection of Online Sexism (Kirk et al., 2023), proposed three tasks, of which two will be discussed in this paper:

- **TASK A** - Binary Sexism Detection: a two-class (or binary) classification problem where systems have to predict whether a post is sexist or not sexist.
- **TASK B** - Category of Sexism: for posts that are sexist, a four-class classification problem where systems have to predict one of four categories: (1) threats, (2) derogation, (3) animosity, (4) prejudiced discussions.

In this paper, we aim to build a sexism detection system for Task A and B, which is built in two phases. First, we evaluate which language model performs best on the tasks. Later, we augment linguistic features to the best-performing language model in all possible combinations and benchmark them.

Specifically, we consider BERT (Devlin et al., 2018a), HateBERT (Caselli et al., 2020) and DeBERTa (He et al., 2020). For linguistic features, we consider the ones described in Abburi et al. (2021): Empath, Hurtlex, and PerspectiveAPI. More details on the models and features can be found in the appendix (Section A and B).

From the phase 1 experiments, we observed HateBERT to perform better than BERT and DeBERTa on both tasks. This observation reinforces Gururangan et al. (2020)'s results on further pretrained language models performing better on the downstream tasks. We conduct another chain of experiments where the best performing variant is further pretrained on 2 million unlabeled sentences provided by the task organizers and later finetuned on the downstream tasks. The phase 2 experiments are benchmarked on the continuously pretrained model and the model with regular pretraining.

We observed that the performance of HateBERT was only improved by continuous pretraining for Task B. We conjecture that further pretraining on the task domain helped the model to understand the nuances required in Task B to distinguish different kinds of sexism.

When benchmarked with linguistic features, we observed them to be redundant. It appears that the large language models are already aware of the task-related information that linguistic features are trying to offer. Regardless of continuous pretraining, we see that the features are not adding

[1] Mean is calculated between local development set and codalab development set. Local development set is a 15 percent split of the training data provided.

any noticeable improvements. Overall, our system (named *SKAM*) ranks number 45 out of 84 for Task A and position 39 out of 69 for Task B against the official test set. The code can be found in the GitHub repository.

2 Data

The dataset consists of 14,000 annotated messages and an additional set of 2 million sentences (1 million sentences obtained from Reddit and 1 million sentences obtained from Gab). This additional data will be used to further pretrain our model. More information about the dataset can be found in Kirk et al. (2023)'s work. In addition to the validation dataset on codalab, we create a local development set which is a 15 percent split of the training data provided.

2.1 Preprocessing

In general, the chosen preprocessing methods focus on n-gram substitution, deletion, and fragmentation. The preprocessing pipeline is adapted to the baselines and the pretrained language models.

To prepare the data for our baselines, we first changed all data entries to lowercase. Second, we have chosen to remove non-words. This means that textual fragments such as URLs, interpunction, and non-alphanumeric characters will be removed. Finally, we perform lemmatization and tokenization provided by NLTK[2] and convert text to term frequency or TF-IDF vectors.

Models such as BERT have difficulty dealing with noise, such as data with URLs or hashtags (Kumar et al., 2020). Since the data originates from social media, these types of noise may occur. Therefore, URLs were replaced with a `[URL]` token. We chose to not replace hashtags, since they might provide information about a message, e.g. a possible topic. Therefore, we fragment the hashtags into separate tokens depicting the words expressed. Another preprocessing approach taken was number replacement. Wallace et al. (2019) have shown that BERT-based models have difficulty working with numbers. Therefore, we treat numbers as noise and replace them with `[NUM]` tokens.

3 Methodology

Our system is built in two phases. In the first phase, three different models are benchmarked against Task A and B. The best-performing model will be used for the next phase, which is feature evaluation. During this phase, three different features are used individually and as ensembles and appended to the best-performing model. In addition, we consider further pretraining the best-performing model selected during phase 1.

In this section, three main aspects of our methodology will be discussed: (1) discussion of language models used, (2) descriptions of features utilized and (3) the motivation behind further pretraining the best performing models of phase 1.

3.1 Baselines

For the baseline, two different methods were used. The first baseline is the most frequent classifier, which always predicts an instance as the most frequent class label in the training data. This results in a macro-F1 score of 0.4317 for Task A and a macro-F1 score of 0.1722 for Task B.

As a second baseline, an SVM (Cortes and Vapnik, 1995) is used, in combination with two different features: TF (term frequency) and TF-IDF (term frequency-inverse document frequency) of n-grams.

The baseline that gives the highest macro-F1 score for Task A is the SVM with TF-IDF (0.7460), while the highest macro-F1 score for Task B is 0.4757, which is obtained by the SVM model with TF (see Table 1).

	Task A		Task B	
Baseline	**Macro-F1**	**Accuracy**	**Macro-F1**	**Accuracy**
Majority class	0.4317	0.7595	0.1722	**0.7557**
SVM + TF	0.7410	0.8300	**0.4757**	0.5176
SVM + TF-IDF	**0.7460**	**0.8371**	0.4486	0.5392

Table 1: Baseline model results for Task A and Task B. Best scores are displayed in boldface.

3.2 Feature-Model Integration

To integrate the features into the classification process, we used the feature vectors to indicate the presence of sexism-related topics. If a certain topic was present in the text, the category name would be appended to the input text.

For Empath and Hurtlex, if a feature value was higher than 0, the feature name would be appended to the input text. Since the feature values of PerspectiveAPI are probabilities, rather than (normalized) frequencies, a threshold was set; If a probability is higher than the threshold, the category would be appended to the input text. We experimented

[2] NLTK Documentation

with thresholds ranging from 0.5 to 0.9 in intervals of 0.1. Overall, a threshold of 0.8 had a consistent, positive performance for Task A and Task B (see Table 3) and was thus chosen as the threshold.

3.3 Continuous pretraining

Continuous learning, also known as continuous pretraining, is a method used to refine a machine learning model's performance by further training it on additional data that is related to the target task. This process helps the model better understand and adapt to the nuances of the domain it is working on, leading to improved performance in downstream tasks. Gururangan et al. (2020) showed that domain adaptive pretraining leads to performance gains in downstream tasks. As we observed HateBERT (Caselli et al., 2020) to perform better than BERT (Devlin et al., 2018b) and DeBERTa (He et al., 2020) on the development set (Table 9 & 10), we conjecture that narrowing the model domain more closely to the task will help us improve the model's performance. To test this hypothesis, we further pretrained HateBERT on the 2 million sentences provided for further pretraining.

Since HateBERT is already further pretrained on hate speech data, we limit the number of epochs to 5 and the learning rate to 1.00e-5. We believe that this setting will allow the model to learn domain-specific nuances while not losing out on the previously gained knowledge from earlier pretraining.

4 Results

4.1 Model Selection

BERT, HateBERT, and DeBERTa were each finetuned using batch size (4, 16, 32 and 64) and learning rate (1.00E-04, 5.00E-05 and 1.00E-05). The models are evaluated using the local development set and the competition development set supplied through Codalab, using the mean of the two macro-F1 scores. The overview of all results of the model finetuning can be found in the Appendix (Section B.1, Table 9 and 10).

HateBERT obtained the highest mean macro-F1 score in Task A (0.8298) and Task B (0.6151). Therefore we selected HateBERT as the model to build upon for the remainder of the experiments for Task A.

4.2 Feature Selection

After establishing HateBERT as the best performing model after finetuning, the additional features Empath, PerspectiveAPI, and Hurtlex were added in all possible combinations. The overview of all results of the best-performing model per task, after adding the different additional feature sets can be found in Table 3 in the appendices. The macro-F1 scores in this table are based on evaluation on the local development set.

For Task A, the model without any additional features is the best performing one (macro-F1 score of 0.837) whereas for Task B, the model in combination with Hurtlex obtained the highest macro-F1 score of 0.627.

After calculating the mean of the local macro-F1 and competition macro-F1 scores, the best performing combination is HateBERT without additional features for Task A (0.8331) and HateBERT with Empath and PerspectiveAPI for Task B (0.6220).

4.3 Further Pretraining the best model

The performance of the model with all possible feature combinations was also tested with continuous pretraining. The overview of the results can be found in Table 5. While continuous pretraining improves the local development set performance of some models in Task A, there is a consistent increase in macro-F1 scores for Task B.

We observed continuous pretraining to not improve the macro-F1 score compared to regular pretraining in Task A. The best model with continuous pretraining achieved a mean macro-F1 score of 0.8259, while the one with regular pretraining scored 0.8331.

The best macro-F1 score for the local Task B development set was obtained by the model combined with Hurtlex using continuous pretraining (0.632). Compared to the model using regular pretraining, a small improvement was made (0.627). The best mean macro-F1 score however was achieved by the model without any additional features (0.6246), which obtained a score of 0.617 on the local set and a score of 0.6322 on the competition set.

5 Discussion

5.1 Why linguistic features did not work?

Albeit on a different task, Younus and Qureshi (2020) showed that linguistic features are helpful when they are computed taking context into consideration. Their model performed similar to a vanilla BERT model when the augmented linguistic features does not consider the context. Although the task seems to be different from ours, this observa-

Table 2: Best performing models per task with macro-F1 and accuracy scores.

Task	Features	Pretraining	Model	Batch size	Learning rate	Mean Macro-F1
A	None	Regular	HateBERT	4	1.00E-05	0.8331
B	None	Continuous	HateBERT	4	5.00E-05	0.6246

Features	Threshold	Task A Local macro-F1	Task B Local macro-F1
None		**0.837**	0.616
E		0.822	0.614
H		0.829	**0.627**
P	0.5	0.815	0.596
	0.6	0.83	0.583
	0.7	0.818	0.574
	0.8	*0.829*	0.616
	0.9	0.823	0.61
H + E		0.834	0.615
P + H	0.8	0.836	0.608
E + P	0.8	0.83	0.608
E + H + P	0.8	0.827	0.575

Table 3: HateBERT scores with added feature combinations for Task A and Task B. The threshold column refers to the used threshold value for PerspectiveAPI. **E** = Empath, **H** = Hurtlex, **P** = PerspectiveAPI

Pretraining	Features	Threshold	Task A Macro-F1	Task B Macro-F1
Regular	None		**0.829**	0.616
	E		0.805	0.582
	H		0.811	0.615
	P	0.8	0.814	0.609
	H + E		0.820	0.599
	P + H	0.8	0.802	0.616
	E + P	0.8	0.815	**0.635**
	E + H + P	0.8	0.828	0.605
Continuous	None		0.820	0.632
	E		0.829	0.613
	H		0.826	0.592
	P	0.8	0.822	0.606
	H + E		0.820	0.608
	P + H	0.8	0.828	0.584
	E + P	0.8	0.828	0.626
	E + H + P	0.8	0.829	0.564

Table 4: Codalab competition results of models with added feature combinations for Task A and Task B, with regular and continuous model pretraining. **E** = Empath, **H** = Hurtlex, **P** = PerspectiveAPI

tion supports our findings and provides a justification for why linguistic features did not work in our experiments.

We found HateBERT to be the best performing model in phase 1, which is specifically pretrained on hate speech texts. This raises the question if it is already tuned to extract the information which extrinsic linguistic features aim to extract.

Koufakou et al. (2020) had a similar approach on extending BERT model with Hurtlex; HurtBERT. When tested on multiple datasets, it improved the baseline in 4 out of 6 cases. We speculate that the current dataset is one of the cases where HurtBERT did not work. It would be interesting to expand HurtBERT with Empath and PerspectiveAPI features and benchmark on the same six datasets.

In this direction, we tried the features in isolation with SVM on Task A. Table 11 shows the scores obtained on the local development set. It can be seen that Hurtlex features outperform PerspectiveAPI & Empath features. Moreover, PerspectiveAPI and Empath features perform similar to Majority Class classifier (Table 11) with an macro-F1 score of 0.43. This shows that Hurtlex features tend to be more informative than other features. It can also observed that adding PerspectiveAPI, Empath or both to Hurtlex features does not improve the model's performance. It is possible that PerspectiveAPI and Empath features are redundant when used with Hurtlex. However, it would be interesting if the same theory holds when tried with HurtBERT on the six datasets mentioned in Koufakou et al. (2020).

5.2 Why Continuous Pretraining worked in one task but not the other?

From the results we found that continuous pretraining was beneficial for HateBERT on Task B, but not on Task A. As Task B is more complex, the model needs to have a better understanding of nuances in the text. Task A is less complex and presumably requires less nuance capturing. We conjecture that further pretraining on the task domain helped the model to understand the nuances required in Task B to distinguish different kinds of sexism.

Features	Threshold	Task A Local macro-F1	Task B Local macro-F1
None		**0.832**	0.617
E		0.828	0.624
H		0.815	**0.632**
P	0.8	0.828	0.619
H + E		0.802	0.615
P + H	0.8	0.823	0.631
E + P	0.8	0.819	0.612
E + H + P	0.8	0.798	0.615

Table 5: HateBERT scores for models with added feature combinations for Task A and Task B with continuous pretraining. The threshold column refers to the used threshold value for PerspectiveAPI. **E** = Empath, **H** = Hurtlex, **P** = PerspectiveAPI

5.3 Error Analysis

The predicted and true labels were compared to make confusion matrices shown in Figure 1, 2, and 3 in the Appendices. While 'not sexist' has more entries in the Task A test and development set, the amount of misclassifications is slightly higher compared to the sexist class (306 ‘not sexist’ vs. 223 ‘sexist’ for the test set, 128 ‘not sexist’ vs. 124 ‘sexist’ for the development set). The proportion of misclassifications is therefore greater for the sexist class in both cases, which indicates a model bias towards the majority class (‘not sexist’).

The confusion matrix for the Task B development set shows misclassifications across the categories. Most misclassifications happen between categories ’2. derogation’ and ’3. animosity’. ‘Aggressive and emotive attacks’ are for example quite likely to also contain ‘gendered slurs, profanities, and insults’ - yet the former subclass belongs to the derogation category and the latter to the animosity category. This demands the need for a well designed system which can learn the nuances between these two classes, re-assessing the structure of the classification problem.

5.4 Feature Integration

During the feature extraction and integration, we represented each collection of feature values as numerical vectors. There are two options to integrate these vectors into the classification process: (1) appending the vector as text to the input, or (2) appending a numerical vector to the output of the classification layer of the classifier model.

While the first approach was chosen, we found that generally, the transformer-based language models did not improve with this implementation of features. Therefore, future research could entail utilizing the numerical nature of the feature vectors in the classification process.

6 Conclusion

In this work, we evaluated the impact of different linguistic features, Hurtlex, PerspectiveAPI and Empath, and found that they are redundant when used with a domain specific language model like HateBERT. We also evaluated if continuous pretraining helps the model in the downstream tasks. We observed it to be beneficial in task B which is more complex and demands more nuanced understanding of the text. As future work, we aim to incorporate context guided linguistic features and experiment with the way we integrate features.

7 Acknowledgements

This paper is the outcome of the course "Shared Task for Information Science" of the Information Science Master. We would like to thank Tommaso Caselli and Lukas Edman for their supervision. One of the authors is supported by the Erasmus Mundus Master Programme in“Language and Communication Technologies”.

References

Charika Abburi, Shradha Sehgal, Himanshu Maheshwari, and Vasudeva Varmak. 2021. Knowledge-based neural framework for sexism detection and classification.

Hind Saleh Alatawi, Areej Alhothali, and Kawthar Moria. 2021. Detection of hate speech using BERT and hate speech word embedding with deep model. *CoRR*, abs/2111.01515.

Elisa Bassignana, Valerio Basile, and Viviana Patti. 2018. Hurtlex: A multilingual lexicon of words to hurt. In *5th Italian Conference on Computational Linguistics, CLiC-it 2018*, volume 2253, pages 1–6. CEUR-WS.

Tommaso Caselli, Valerio Basile, Jelena Mitrovic, and Michael Granitzer. 2020. Hatebert: Retraining BERT for abusive language detection in english. *CoRR*, abs/2010.12472.

Corinna Cortes and Vladimir Vapnik. 1995. Support-vector networks.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018a. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2018b. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Ethan Fast, Binbin Chen, and Michael S. Bernstein. 2016. Empath: Understanding topic signals in large-scale text. *CoRR*, abs/1602.06979.

Suchin Gururangan, Ana Marasovic, Swabha Swayamdipta, Kyle Lo, Iz Beltagy, Doug Downey, and Noah A. Smith. 2020. Don’t stop pretraining: Adapt language models to domains and tasks. *CoRR*, abs/2004.10964.

Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. 2020. Deberta: Decoding-enhanced BERT with disentangled attention. *CoRR*, abs/2006.03654.

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*, Toronto, Canada. Association for Computational Linguistics.

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.

Ankit Kumar, Piyush Makhija, and Anuj Gupta. 2020. Noisy text data: Achilles' heel of BERT. In *Proceedings of the Sixth Workshop on Noisy User-generated Text (W-NUT 2020)*, pages 16–21, 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. *CoRR*, abs/1907.11692.

Athar Hussein Mohammed and Ali H. Ali. 2021. Survey of bert (bidirectional encoder representation transformer) types. *Journal of Physics: Conference Series*, 1963(1):012173.

Eric Wallace, Yizhong Wang, Sujian Li, Sameer Singh, and Matt Gardner. 2019. Do nlp models know numbers? probing numeracy in embeddings. *arXiv preprint arXiv:1909.07940*.

Arjumand Younus and M Atif Qureshi. 2020. Combining bert with contextual linguistic features for identification of propaganda spans in news articles. In *2020 IEEE International Conference on Big Data (Big Data)*, pages 5864–5866. IEEE.

A Models

BERT is a widely used pretrained language model and obtains good results in different tasks (Alatawi et al., 2021). Since the introduction of BERT, some modified versions have succeeded in providing a new state-of-the-art (Mohammed and Ali, 2021). Regardless of modified versions, BERT has shown good performance in text classification tasks due to its bidirectional pretraining.

DeBERTa is a successor of BERT which uses disentangled attention mechanism and an enhanced mask decoder. Both techniques aid in classification by placing emphasis on the position of a word, and not only taking into account the content itself. Trained on only half the data used for RoBERTa-Large (Liu et al., 2019), He et al. (2020) were able to provide competitive performance. Due to its context-sensitive techniques, DeBERTa is expected to be useful for this task.

HateBERT is a version of BERT that's further pretrained on a large Reddit hate speech data set. It outperformed BERT in three different evaluation sets (OffensEval, AbusEval, HatEval), showing versatility in terms of practical applications. HateBERT is expected to obtain good results due to its close relation to the task data.

B Features

For this task, we will utilize three feature extraction approaches as suggested by Abburi et al. (2021): (1) Empath, (2) PerspectiveAPI, and (3) HurtLex. Abburi et al. (2021) illustrate that these approaches can be used to extract additional linguistic and semantic knowledge from textual data, and to aid in the classification of sexism. Therefore, we hypothesize that the employment of these feature extraction methods could improve the performance of the BERT-based models within the context of the explainable detection of sexism.

Empath is a tool for analyzing text across lexical categories (Fast et al., 2016). It draws connotations between words and phrases by deep learning across more than 1.8 billion words of modern fiction and can generate new lexical categories. Abburi et al. (2021) suggested the usage of a 21-dimensional feature vector with each value depicting the presence of a concept related to sexism (for individual features, see Appendix C). If a text contains a unigram related to one of the categories, that category would be assigned a normalized frequency value.

Google's PerspectiveAPI[3] is a machine learning algorithm that measures the effect of a text by analyzing different emotional concepts. The PerspectiveAPI features are represented by a vector consisting of nine probabilities. The categories corresponding to each value are in alphabetical order (see Appendix C). For each text, the API provides a 9-dimensional vector with values ranging from 0 to 1 related to the corresponding emotional concept.

HurtLex is a lexicon that includes aggressive, offensive, and hateful words and phrases divided into seventeen categories (Bassignana et al., 2018). From the seventeen categories, the nine categories from Abburi et al. (2021) were used, as well as an additional tenth category 'male genitalia' (see Appendix C). The texts will be represented by this 10-dimensional feature vector based on frequencies.

B.1 Model Selection: Hyperparameter search and experiments

BERT, HateBERT, and DeBERTa were each finetuned using batch size (4, 16, 32 and 64) and learning rate (1.00E-04, 5.00E-05 and 1.00E-05). The models are evaluated using the local development set and the competition development set supplied through Codalab, using the mean of the two macro-F1 scores. The overview of all results of the model finetuning can be found in Table 9 and 10 in the Appendix.

DeBERTa and HateBERT both scored considerably better than BERT on the local development set. DeBERTa with batch size 32 and a learning rate of 1.00E-05 obtained a higher macro-F1 score (0.839) compared to HateBERT with batch size 4 and a learning rate of 1.00E-05 (0.834) and BERT with batch size 32 and a learning rate of 1.00E-05 (0.818). On the competition development set, HateBERT obtained a higher macro-F1 score (0.8255) compared to DeBERTa (0.8173).

HateBERT obtained the highest mean macro-F1 score (0.8298) and was therefore selected as the model to build upon for the remainder of the experiments for Task A.

A similar score distribution can be observed in Task B, where BERT obtained a lower score compared to DeBERTa and HateBERT. On the competition development set HateBERT outperformed DeBERTa with a macro-F1 score of 0.6161 compared to 0.5589. The highest mean macro-F1 score was

[3] https://perspectiveapi.com/

obtained by HateBERT (0.6151 compared to DeBERTa's 0.5855), which is therefore also selected as the model base for Task B experiments.

Label	**Percentage**	**Amount**
Not Sexist	76%	10,602
Sexist	24%	3,398

Table 6: Label distribution of the data for Task A

Label	**Percentage**	**Amount**
Derogation	47%	1,590
Animosity	34%	1,165
Prejudiced Discussion	10%	333
Threats	9%	310

Table 7: Label distribution of the data for Task B

C Feature Categories

Table 8: Overview of features used from 3 feature extraction approaches: (1) Empath, (2) PerspectiveAPI and (3) HurtLex

Empath	**PerspectiveAPI**	**HurtLex**
Sexism	Flirtation	Negative stereotypes and ethnic slurs (PS)
Violence	Identity attack	Professions and occupations (PA)
Money	Insult	Physical disabilities and diversity (DDF)
Valuable	Obscene	Cognitive disabilities and diversity (DDP)
Domestic work	Profanity	Female genitalia (ASF)
Hate	Severe Toxicity	Words related to prostitution (PR)
Aggression	Sexually explicit	Words related to homosexuality (OM)
Anticipation	Threat	With potential negative connotations (QAS)
Crime	Toxicity	Derogatory words (CDS)
Weakness		Male genitalia (ASM)
Horror		
Swearing terms		
Kill		
Sexual		
Cooking		
Exasperation		
Body		
Ridicule		
Disgust		
Anger		
Rage		

D Finetuning models Task A

Table 9: Finetuning scores for different models in Task A on the development set. Best macro-F1 and accuracy scores per model are displayed in boldface. Best performing model scores overall are underlined.

Model	Batch size	Learning rate	Macro-F1	Accuracy
BERT	32	1.00E-04	0.808	0.863
		5.00E-05	0.801	0.842
		1.00E-05	**0.818**	0.858
	16	1.00E-04	0.789	0.85
		5.00E-05	0.811	0.85
		1.00E-05	0.809	**0.865**
	4	1.00E-04	0.196	0.244
		5.00E-05	0.196	0.244
		1.00E-05	0.813	0.862
HateBERT	32	1.00E-04	0.81	0.852
		5.00E-05	0.829	0.873
		1.00E-05	0.831	0.874
	16	1.00E-04	0.782	0.842
		5.00E-05	0.834	0.87
		1.00E-05	0.824	0.871
	4	1.00E-04	0.196	0.244
		5.00E-05	0.792	0.835
		1.00E-05	**0.837**	**<u>0.883</u>**
DeBERTa	64	1.00E-04	0.802	0.843
		5.00E-05	0.815	0.853
		1.00E-05	0.825	0.86
	32	1.00E-04	0.431	0.756
		5.00E-05	0.82	0.862
		1.00E-05	**<u>0.839</u>**	**0.879**
	16	1.00E-04	0.196	0.244
		5.00E-05	0.196	0.244
		1.00E-05	0.834	0.871

E Finetuning models Task B

Table 10: Finetuning scores for different models in Task B on the development set. Best macro-F1 and accuracy scores per model are displayed in boldface. Best performing model scores overall are underlined.

Model	Batch size	Learning rate	Macro-F1	Accuracy
BERT	32	1.00E-04	0.577	0.568
		5.00E-05	0.575	0.578
		1.00E-05	**0.59**	**0.586**
	16	1.00E-04	0.503	0.533
		5.00E-05	0.59	0.569
		1.00E-05	0.581	0.574
	4	1.00E-04	0.045	0.1
		5.00E-05	0.56	0.564
		1.00E-05	0.576	0.576
HateBERT	32	1.00E-04	0.613	0.586
		5.00E-05	0.606	0.598
		1.00E-05	0.601	0.584
	16	1.00E-04	0.582	0.561
		5.00E-05	0.59	0.582
		1.00E-05	0.587	0.584
	4	1.00E-04	0.045	0.1
		5.00E-05	**<u>0.614</u>**	**0.607**
		1.00E-05	0.601	0.584
DeBERTa	64	1.00E-04	0.125	0.334
		5.00E-05	0.609	**<u>0.617</u>**
		1.00E-05	0.505	0.508
	32	1.00E-04	0.585	0.564
		5.00E-05	0.581	0.561
		1.00E-05	0.611	0.615
	16	1.00E-04	0.046	0.1
		5.00E-05	0.601	0.598
		1.00E-05	**0.612**	0.592

F Feature scores

Table 11: Accuracy and macro-F1 Score for different feature combinations
E = Empath, **H** = Hurtlex, **P** = PerspectiveAPI

Input	Task-A Accuracy	Task-A Macro-F1	Task-B Accuracy	Task-B Macro-F1
H	0.8057	0.62	0.7786	0.25
P	0.7562	0.43	0.7562	0.17
E	0.7562	0.43	0.7562	0.17
H + E	0.8057	0.62	0.7786	0.25
P + E	0.7562	0.43	0.7562	0.17
P + H	0.8057	0.62		
P + H + E	0.8057	0.62		

G Confusion matrix - Task A test set

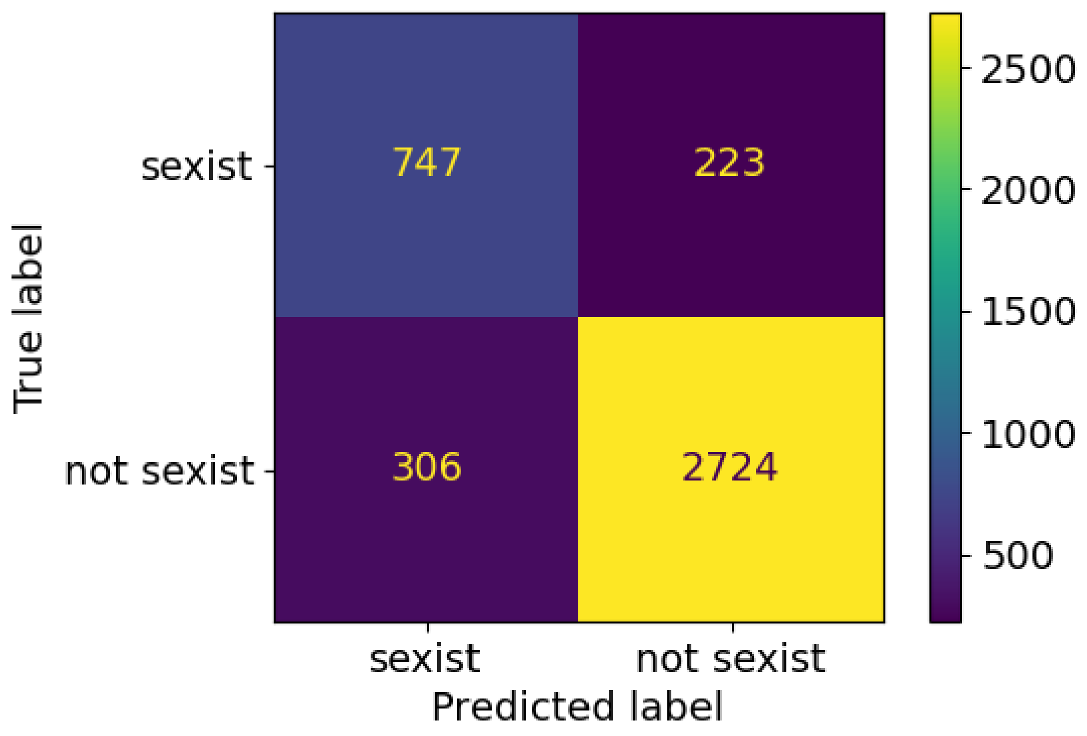

Figure 1: Confusion matrix of predictions and true labels for Task A, competition test set.

H Confusion matrix - Task A development set

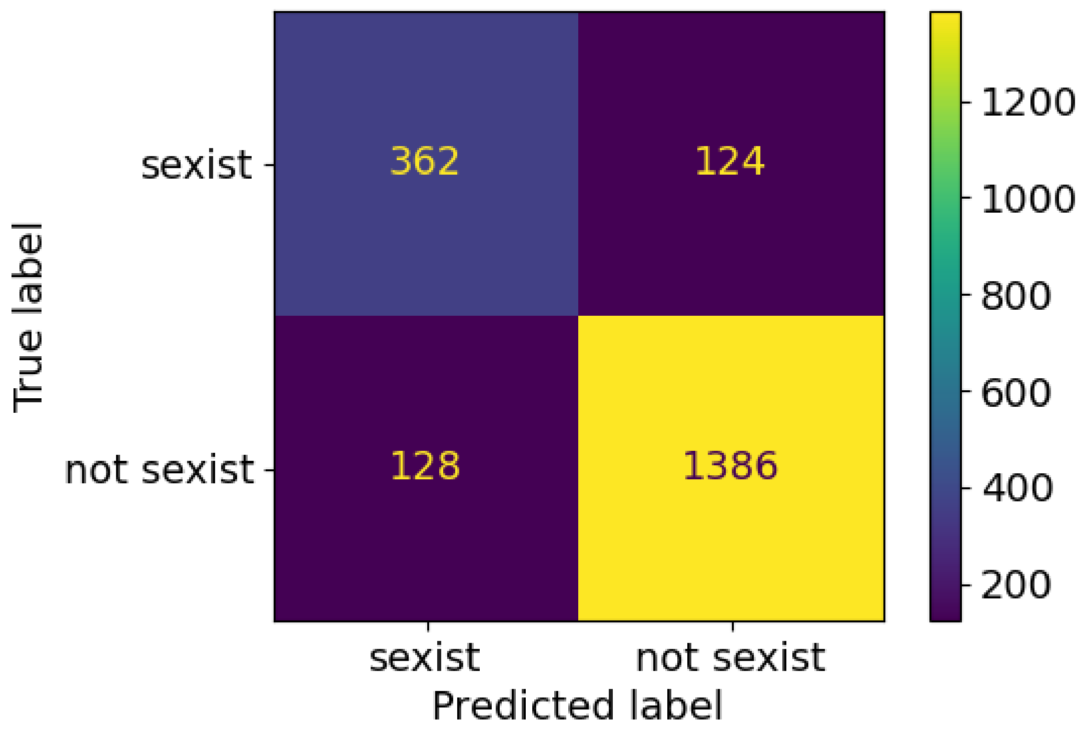

Figure 2: Confusion matrix of predictions of true labels for Task A, competition development set.

I Confusion matrix - Task B development set

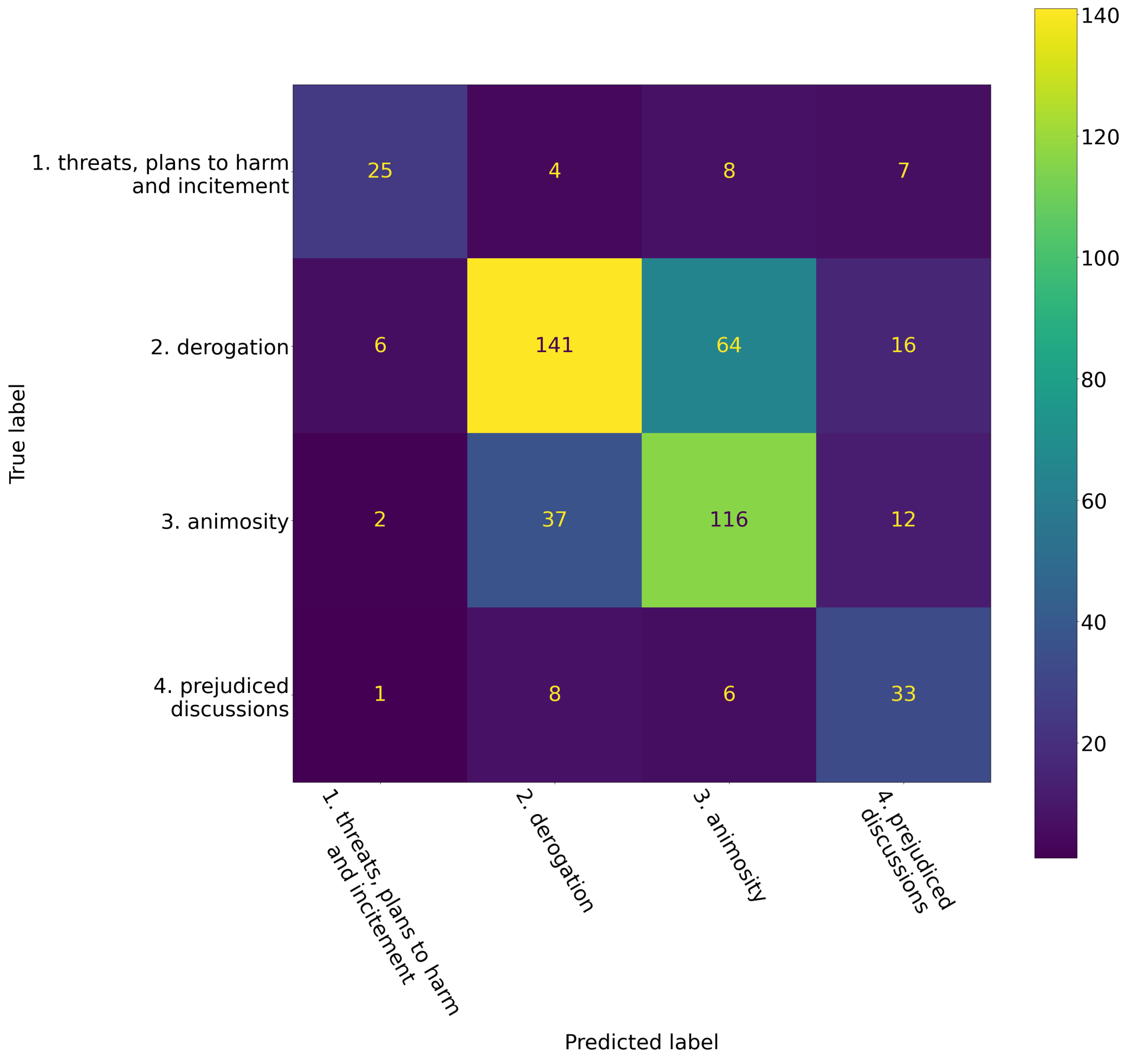

Figure 3: Confusion matrix of predictions of true labels for Task B, competition development set.