Team Quabynar at the GermEval 2024 Shared Task 1 GerMS-Detect (Subtasks 1 and 2) on Sexism Detection

Kwabena Odame Akomeah
University of Regensburg
kwabena-odame.akomeah@ur.de

Udo Kruschwitz
University of Regensburg
udo.kruschwitz@ur.de

Bernd Ludwig
University of Regensburg
bernd.ludwig@ur.de

Abstract

While large language models such as ChatGPT and GPT-3.5 Turbo offer impressive capabilities, their use can be costly and may not always be advisable, particularly for specific types of tasks. As part of our involvement in the GerMS-Detect challenge we observe that traditional, more cost-effective language models such as BERT are able to achieve better results than GPT-3.5 Turbo, a very robust LLM, when applied to sexist text classification. This suggests that for certain types of tasks and contexts, using BERT models may be a plausible alternative to state-of-the-art LLMs. This paper hightlights our approach to predicting annotator binary and soft labels using transformer models and an LLM in GermEval 2024 GerMS-Detect's open and closed subtasks of sexism detection.

1 Introduction

In an era where social media conversations and text proliferates exponentially (Guo et al., 2022), the need for vigilant moderation has gained much attention. As more people engage online, the challenge of identifying hate speech, toxic comments, and other harmful content has become increasingly urgent (Ayele et al., 2023). While much research has focused on English, the impact of harmful content extends beyond language barriers (Jahan and Oussalah, 2023). The link between hate speech and the spread of sexism is profound as the former can be explained in the later (Sen et al., 2022). Hate speech often perpetuates negative stereotypes and harmful ideologies, reinforcing societal norms that can lead to marginalization and oppression (Richardson-Self, 2021). This creates a feedback loop where sexist attitudes are normalized and disseminated through various channels, including social media, public discourse, and interpersonal interactions (Fox et al., 2015). The impact of this dynamic is far-reaching, influencing not only individual behaviors and beliefs but also institutional policies and cultural narratives (Richardson-Self, 2021). Understanding this connection is crucial for developing effective strategies to combat both hate speech and sexism, promoting a more inclusive and equitable society. Sexism can incite targeted hate or even violence against specific groups based on sex orientatation and identification (Sen et al., 2022). Thus, identifying content that warrants scrutiny is as crucial as identifying outright hate speech.

GermEval 2024 GerMS-Detect is part of a series of shared task evaluation campaigns that focus on Natural Language Processing (NLP) for the German language. This year's GerMS-Detect subtasks specifically target sexism detection in German online news fora, building upon previous years' efforts on detecting various texts with hate speech (Risch et al., 2021). GerMS-Detect goes beyond toxicity identification. It also delves into classifying annotated data by several annotators and combining them in different ensemble formats to attain both binary and soft labels aiming to foster healthier conversations online.

The goal of **subtask 1** was to predict labels for each text in a dataset, with these labels derived from the original annotations made by multiple human annotators. In contrast, **subtask 2** sought to predict the label distribution for each text in the same dataset, with this distribution based on the original allocation of labels predicted in subtask 1. These two subtasks were interrelated, as both aimed to accurately reflect the human annotators' evaluations, with subtask 1 focusing on discrete label prediction and subtask 2 on capturing the nuanced distribution of these labels.

Our participation in GermEval 2024 involved tackling two subtasks both in the closed and open competitions. Leveraging transformer-based model architectures from the huggingface repository, we specifically employed different BERT embeddings and finetuned with Pytorch for the closed compe-

tition and an LLM for the open competition. In the following sections, we will examine the dataset employed, discuss the selected model architectures in detail, and evaluate their performance on the designated subtasks. To support reproducibility the complete codebase for our experiments is available as a Github repository [1]

The dataset used in GerMS-Detect were delivered sequentially in 3 different sections namely the trial, development and competition phases where the later was a consolidation of all the dataset from the other two phases. The data used for training and testing with 5,998 labeled texts for training and 1,986 unlabeled for testing annotated by 10 human annotators sporadically. The distribution of annotations is uneven across the annotators. Three annotators (A002, A012, A010) have annotated all 5,998 texts, while others have annotated fewer, with the least being A001, who annotated 970 texts. Submissions were made on Codabench [2]

2 Model architectures

As set out in the shared task guidelines, our approach for the closed competition exclusively employed untrained transformer models that had not been exposed to any sexism or hate speech data. The models utilized in our study included Google's German BERT cased, multilingual BERT and Deepset's German BERT base.

General Approach. The data was loaded and preprocessed by aggregating all annotators into a unified dictionary of separate annotator dataframes. This new dataframe constituted the columns, ids, text and labels labels texts for each annotator in the dictionary. The initial data which was in JSON format had lists of annotators in one column and labels in another for each distinct id and text.

```
{'A001':
        id   \

 0     a733e8a47708ce1d77060266d365e5b5
 1     bf45fc2ac6742a7f75d5863c3338d59d
 2     e1e80ff680f874d49ddfe33ac846a454
 3     4689b9ccb5d79f222ba110f389cf1fb6
 4     a8d04dfc8e63b67f4587b04524605e3e
 ..                                 ...
 965   b13f0c202385d74b54f1fac4ea297510
 966   f3bc2e041355ab9c1bba465a004f0631
 967   841b039088a4edc7a14df7b231fd2f85
 968   2414c1c9fd116539262aba5ee58de650
 969   075cfd6f6dacfb11cc0a919bb21d70d2

      text
0     Wen man nicht reinlä ß t,...
1     Und eine Katze die schnurrt...
2     Des Oaschloch is eh scho ...
3     Trump hat 2 Dinge übersehen:...
4     Mit der Fox÷e hat er sich ...
..
965   was hat Zadic dazu veranlasst...
966                Uninteressant.
967   dem Islam die Frauenfeindl...
968   vielleicht spitzt sie jetzt...
969   Die Geschichte mit Astra...

     label  label_text
0      0        -Kein
1      0        -Kein
2      0        -Kein
3      0        -Kein
4      4      -Extrem
..
965    3       -Stark
966    0        -Kein
967    0        -Kein
968    0        -Kein
969    2   -Vorhanden
[970 rows x 4 columns],
'A002':
        id   \
0     a733e8a47708ce1d77060266d365e5b5
1     bf45fc2ac6742a7f75d5863c3338d59d
2     e1e80ff680f874d49ddfe33ac846a454
3     4689b9ccb5d79f222ba110f389cf1fb6
4     a8d04dfc8e63b67f4587b04524605e3e
...
      text
0     Wen man nicht reinlä ß t,...
1     Und eine Katze die schnurrt...
2     Des Oaschloch is eh scho ...
3     Trump hat 2 Dinge übersehen:...
4     Mit der Fox÷e hat er sich...

       label label_text
0          0      -Kein
1          0      -Kein
2          3     -Stark
3          3     -Stark
4          4    -Extrem
...
[5998 rows x 4 columns],
... }
```

Listing 1: A sample of the dictionary of annotator dataframes

Subsequently, the texts within the JSON file were regrouped by annotators and placed in container dataframes. The dataset was then partitioned into training and validation sets using a 90/10 percentage split for each annotator dataframe. This reason for ratio of 90:10 was ensure that majority of the dataset was included in the training rather than validation.

The dataset was then tokenized using the various BERT tokenizers implemented under the API Transformers and prepared for training with a batch

[1] https://github.com/kaodamie/Quabynar--GermDetect-2024

[2] https://www.codabench.org/competitions/

size of 16, a threshold that was appropriate for the GPU memory available. It is important to note that different BERT models have their corresponding tokenizers and it is advisable to stick to a matching tokenizer as this can affect training significantly. The BERT tokenizer is a tool that processes and converts input text into tokens, which are the basic units of text that the BERT model can use for various NLP tasks (Devlin et al., 2019). The BERT tokenizer is based on a specific tokenization technique called WordPiece (Devlin et al., 2019). Due to the training structure, which involved looping over several annotators, a significant amount of memory was required. With a GPU RAM of 12GB running PyTorch CUDA 12.0, the memory constraints were adequately managed.

Throughout the training process, metrics including accuracy, precision, recall, and F1 score were monitored and recorded for each iteration. The training loop was designed to iteratively adjust the model parameters, optimizing the learning rate and minimizing the loss function. Additionally, early stopping criteria with a patience of 3 epochs was implemented to prevent overfitting, ensuring the model maintained its generalization capability on the validation set and to save time taken to train the dataset. The model only stops and saves best weights when training metrics being tracked for each epoch does not improve after 3 additional epochs.

Post-training, the model's performance was evaluated using the unseen test data to confirm its robustness and reliability on codabench as per the regulations of the shared tasks. This was the only testing done due to the dataset size. Further splitting of the training dataset would have resulted in a smaller size for training that would have impacted the training results and model's capabilities. We observed that with this approach, testing results did what was achieved during training.

Subtask 1: Binary and Multi Labels Strategy

The objective of subtask 1 was to predict labels for each text in a dataset, where these labels are derived from those initially assigned by multiple annotators. The annotators, following certain annotation guidelines (Krenn et al., 2024), evaluated the presence and intensity of misogyny or sexism in the texts using the following labels:

- **0-Kein**: No sexism or misogyny present
- **1-Gering**: Mild sexism or misogyny
- **2-Vorhanden**: Sexism or misogyny present
- **3-Stark**: Strong sexism or misogyny
- **4-Extrem**: Extreme sexism or misogyny

Consequently, the degree of strength assigned to a text deemed sexist is largely subject to the annotator's personal judgment. Subtask 1 involved predicting labels based on various strategies for aggregating the multiple annotations into a single target label. The strategies are as follows:

- **bin_maj**: Predict 1 if the majority of annotators assigned a label other than 0-Kein. Predict 0 if the majority assigned a label of 0-Kein. If there is no clear majority, both labels 1 and 0 are accepted for evaluation.
- **bin_one**: Predict 1 if at least one annotator assigned a label other than 0-Kein, and 0 otherwise.
- **bin_all**: Predict 1 only if all annotators assigned labels other than 0-Kein, and 0 otherwise.
- **multi_maj**: Predict the majority label. If no majority label exists, any of the assigned labels are considered correct for evaluation.
- **disagree_bin**: Predict 1 if there is any disagreement among annotators and 0 otherwise.

The strategy for **multi_maj** was slightly modified to predict non-zero labels which was useful on the test data if there was no clear majority as follows: **multi_maj**: Predict the majority label. If no majority label exists, if non-zero labels exist, any of the assigned labels that are **non-zeros** are considered correct for evaluation otherwise predict zero. It was observed to be a more effective approach to attaining a much higher score on the leader-boards for both subtasks.

After this only text ids with the predictions of the test set were saved in .tsv format and uploaded for scoring.

Subtask 2: Predicting Annotator Distributions

For the subtask of predicting annotator distributions, only 2 runs were submitted to achieve a high score of 0.29 over the Shannon-Jensen evaluation

of soft labels topping the leaderboard on subtask 2. The models employed in subtask 2 were Google's BERT mulitlingual cased and German BERT cased model. They were both fine-tuned over a GPU with a RAM of 12GB using PyTorch CUDA 12.0 with the same hyperperparameters discussed in subtask 1. In effect, the models were trained once for both subtasks and binary labels attained in subtask 1 were consequently converted to soft labels to fit subtask 2. Two types of distributions considered are binary score and multi-score distributions. Each set of distribution summed to 1.

- **dist_bin_0**: Represents the proportion of annotators who labeled the text as 'not-sexist' (0-Kein).
- **dist_bin_1**: Represents the proportion of annotators who labeled the text as 'sexist', which includes any of the labels 1-Gering, 2-Vorhanden, 3-Stark, or 4-Extrem.
- **dist_multi_0**: The proportion of annotators labeling the text as 0-Kein.
- **dist_multi_1**: The proportion of annotators labeling the text as 1-Gering.
- **dist_multi_2**: The proportion of annotators labeling the text as 2-Vorhanden.
- **dist_multi_3**: The proportion of annotators labeling the text as 3-Stark.
- **dist_multi_4**: The proportion of annotators labeling the text as 4-Extrem.

The strategy for prediction involved basing the distributions described above on the predicted binary labels from subtask 1 and applying the rules defined for substask 2 in subtask 2. This approach achieved a final score of 0.29 . See Table 2

All models were trained using the AdamW optimizer, initialized with the model's parameters and a learning rate of 2e-5. AdamW is a variant of the Adam optimizer that includes weight decay for better regularization. The loss was computed from the model's outputs. This loss quantifies how well the model's predictions match the target values. These were calculated over the accuracy of logits. Precision, recall and weighted F1 scores were carefully monitored per each epoch run for each model. The backward loss computed dloss/dx for every parameter x. These are accumulated into a gradient variable for every parameter x. The optimizer then updated the value of x using the gradient calculated above. This process was achieved by setting parameters using the "BertForSequenceClassification" definition in the transformer API. It was applied for each annotator in a loop, and the best weights were selected with early stopping over a total of 10 epochs, with a patience threshold of 3 epochs per annotator. See documentation at Huggingface.[3]

3 Open Subtask- Improving Training and Considerations for LLMs

The open competition was permitted for both subtasks 1 and 2 as specified in the terms and agreement of the competition. They allowed for models that had already been pretrained on sexism data and the use of additional dataset provided all information are open-source and results can be replicated. It was unrestricted and open to the use of models such as LLMs.

We applied few-shot learning on OpenAI's GPT 3.5 Turbo. We designed our model to select only the top 5 entries iteratively for each annotator. The model is also designed to preprocess a given text by truncating it to a specified length of 512 characters and then generate a short-length prediction using the GPT-3.5-turbo model with a 5 learning prompt. The truncate procedure ensures the input text remains within the length constraints, while a generate prediction function constructed an appropriate prompt and makes an API call to the language model to obtain a prediction. The parameters used were as follows:

Initialization. An instance of the OpenAI client is created using a provided API key. This client will be used to interact with the OpenAI API for generating text completions. Usage of OpenAI's models come at a cost based on the model and total tokens queried into the API.

Prompting. Prompting was conducted in English while instructing the LLM to analyze and respond to texts in German for the selected few-shot examples. The inclusion of the prompt message *"You are a helpful assistant."* in the API call serves to establish the context for the model, directing it to adopt a cooperative and helpful tone throughout the interaction.

This preliminary instruction is crucial as it sets

[3] https://huggingface.co/transformers/v3.0.2/model_doc/bert.html

Model	bin maj f1	bin one f1	bin all f1	multi maj f1	disagree bin f1	Final Score
BERT Multilingual Cased	0.6579	0.7158	0.5091	0.2393	0.6022	0.5448
German BERT Cased	0.6698	0.7344	0.6091	0.3423	0.6175	0.5946
German BERT Base	**0.6981**	**0.7290**	**0.6428**	**0.3706**	**0.6139**	**0.6109**
***GPT 3.5 Turbo**	*0.4751	*0.5586	*0.4849	*0.1967	*0.5455	*0.4522

Table 1: Subtask 1 Scores of Models. *Subtask 1 Open Results. Scores in green are our best scores attained for subtask 1

the expectation for the assistant's behavior, aiming to enhance the overall quality and relevance of the responses. By providing a clear and explicit directive on the assistant's role, the model is better positioned to deliver accurate, contextually appropriate, and user-friendly outputs, thereby improving the effectiveness of the interaction. This approach ensures that the assistant not only processes the given examples accurately but also maintains a consistent and supportive demeanor in its responses.

```
def generate_prediction(text, few_shot_prompt, max_tokens=50):
    text = truncate_text(text)  # Truncate the text to avoid exceeding the context length
    prompt = few_shot_prompt + f"Text: {text}\nLabel: "
    response = client.chat.completions.create(
        model="gpt-3.5-turbo",
        messages=[
            {"role": "system", "content": "You are a helpful assistant."},
            {"role": "user", "content": prompt}
        ]
...

def format_few_shot_prompt(examples, max_examples=5):
    prompt ="The following texts are in German. Assign a label from 0 (not offensive) to 4 (most offensive):\n\n"
    for example in examples[:max_examples]:
        prompt += f"Text: {example['text']}\nLabel: {example['label']}\n\n"
    return prompt
```

Listing 2: Code snippet

API Parameters. These parameters (max tokens as 50, n as 1 and a temperature of 0.3) are used in the API call to fine-tune the behavior and output of the model to best fit the task of classifying the offensiveness of the texts. By limiting the response to 50 tokens, we ensure that the output is concise and stays within a manageable length. This helps to control the cost and speed of the API call, as well as to avoid overly verbose answers. A lower temperature value (closer to 0) makes the output more deterministic and focused, while a higher value (closer to 1) makes it more random and creative. Setting the temperature to 0.3 made the model to provide more consistent and reliable predictions, which is generally suitable for tasks requiring accuracy. By setting n to 1, the function requests a single response from the model. This is useful for straightforward tasks where only one prediction is needed. If n were greater than 1, multiple responses would be generated, which could be useful for tasks requiring multiple perspectives or for ensemble methods.

Although the LLM's results were not actively explored by the team, there is room for much improvement in the technique by which training of the dataset was done. Selecting first 5 examples and sequentially prompting for each annotator in a loop may not necessarily ensure a balanced example text.

Effective prompting strategies and the inherent capabilities of large language models (LLMs) have in recent times been very popular in many different tasks and text generation (Liu et al., 2023). However, for this particular task, there are ways to further enhance the training process and considerations that need to be made regarding the use of LLMs in such future tasks in order to enhance results.

Annotated Data Quality. Ensuring that the data annotated by different users is of high quality and accurately represents the categories being predicted can significantly improve model performance (Li, 2024).

Data Diversity. Increasing the diversity of the training data by incorporating a wider range of examples from different contexts can help the model generalize better to unseen data (Chung et al., 2023). This includes expanding the dataset to cover various dialects, jargon, and situational contexts.

Contextual Prompts. Utilizing more sophisticated prompting techniques that provide better context and clearer instructions can help models like GPT-3.5 generate more accurate responses. Experimenting with different prompt formats and iteratively refining them based on feedback can lead to better outcomes (Zhou et al., 2023).

Hybrid Models. Exploring approaches that combine the strengths of different models, such as using BERT for initial feature extraction and GPT-3.5 Turbo for generating refined predictions, can leverage the advantages of each model type (Veeramani et al., 2024).

Ensemble Methods. Implementing ensemble methods that aggregate predictions from multiple models can enhance robustness and accuracy, particularly when dealing with complex or ambiguous inputs (García-Díaz et al., 2023).

4 Results

The results after prediction were put together indexing the ids of the various texting and joining the annotators and their corresponding predicitons in different sets over 2 columns. This new dataframe was saved as a compressed tsv file which was then submitted on Codabench. The performance of the system on all five predicted labels were evaluated using the F1 macro score across all classes before averaging the results as the final ranking. See Table 1.

The performance in subtask 2 were assessed using the Jensen-Shannon (JS) distance. This evaluation is applied to both the prediction of the binary distribution and the prediction of the multi-score distribution. The final score was determined by taking the unweighted average of the JS distances for both the binary and multi-score distribution predictions. See Table 2

For our final submission, the model fine-tuned on Deepset's German BERT base achieved a score of 0.61 on the test set for the binary subtask 1 (See Table 1) placing third whereas the model fine-tuned on Google's German BERT cased obtained a score of 0.29 over the Shannon-Jensen evaluation topping the leaderboard for subtask 2 on Codabench. See Table 2

Considering the results derived for the test set, the F1 scores for the multi majority were fairly low following an imbalance coverage in annotation by annotators. We believe much better metrics can be achieved given a much balanced dataset. This incomplete coverage suggests that some texts were annotated by only a subset of annotators, leading to potential biases or inconsistencies in the labeling process impacting model's training and testing, as the diversity of annotations for each text varies. These insights highlight the importance of analyzing annotator contributions and ensuring a fair and comprehensive annotation process for more robust ensemble model development in future.

5 Conclusion

While GPT-3.5 Turbo and similar LLMs have shown promise, particularly due to advancements in prompting and tokenization, it is crucial to evaluate their cost-effectiveness and suitability for specific tasks. BERT models, with their strong performance in this subtask, highlight the importance of selecting the right model based on task requirements. Future training improvements that focus on enhancing data quality, refining prompting strategies, optimizing tokenization, and exploring hybrid approaches through the implementation of ensemble methods that aggregate predictions from multiple models can potentially enhance robustness and accuracy, particularly when dealing with complex or ambiguous inputs. Ultimately, the choice of model should be guided by a careful assessment of the task at hand, resource availability, and the desired balance between performance and cost (Yang et al., 2024).

6 Future work

The findings from our current study emphasize several key areas for future work in leveraging large language models (LLMs) like GPT-3.5 Turbo and other top performing LLMs not mentioned in this study. Randomizing and balancing few-shot examples and comparing them accross various LLMs, adopting different fine-tuning approaches to few-shot and ensemble approaches with both LLMs and transformers are worth exploring. By addressing these areas, future work can build on the promising results of current LLMs, leading to more robust, accurate, and cost-effective applications in natural language processing and beyond.

References

Abinew Ali Ayele, Skadi Dinter, Seid Muhie Yimam, and Chris Biemann. 2023. Multilingual racial hate speech detection using transfer learning. In *Proceedings of the 14th International Conference on Recent Advances in Natural Language Processing*, pages

Model	JS Dist Bin	JS Dist Multi	Final Score
BERT Multilingual Cased	0.2646	0.3517	0.3081
German BERT Cased	**0.2479**	**0.3361**	**0.2920**
***GPT-3.5 Turbo**	*0.3661	*0.4521	*0.4091

Table 2: Subtask 2 Evaluation scores for different models. *Subtask 2 Open results. Scores in green are our best scores attained for subtask 2

41–48, Varna, Bulgaria. INCOMA Ltd., Shoumen, Bulgaria.

John Chung, Ece Kamar, and Saleema Amershi. 2023. Increasing diversity while maintaining accuracy: Text data generation with large language models and human interventions. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 575–593, Toronto, Canada. Association for Computational Linguistics.

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.

Jesse Fox, Carlos Cruz, and Ji Young Lee. 2015. Perpetuating online sexism offline: Anonymity, interactivity, and the effects of sexist hashtags on social media. *Computers in human behavior*, 52:436–442.

José Antonio García-Díaz, Camilo Caparros-laiz, Ángela Almela, Gema Alcaráz-Mármol, María José Marín-Pérez, and Rafael Valencia-García. 2023. UMUTeam at SemEval-2023 task 12: Ensemble learning of LLMs applied to sentiment analysis for low-resource African languages. In *Proceedings of the 17th International Workshop on Semantic Evaluation (SemEval-2023)*, pages 285–292, Toronto, Canada. Association for Computational Linguistics.

Zhijiang Guo, Michael Schlichtkrull, and Andreas Vlachos. 2022. A survey on automated fact-checking. *Transactions of the Association for Computational Linguistics*, 10:178–206.

Md Saroar Jahan and Mourad Oussalah. 2023. A systematic review of hate speech automatic detection using natural language processing. *Neurocomputing*, 546:126232.

Brigitte Krenn, Johann Petrak, Marina Kubina, and Christian Burger. 2024. GERMS-AT: A sexism/misogyny dataset of forum comments from an Austrian online newspaper. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 7728–7739, Torino, Italia. ELRA and ICCL.

Jiyi Li. 2024. A comparative study on annotation quality of crowdsourcing and llm via label aggregation. In *ICASSP 2024-2024 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 6525–6529. IEEE.

Pengfei Liu, Weizhe Yuan, Jinlan Fu, Zhengbao Jiang, Hiroaki Hayashi, and Graham Neubig. 2023. Pre-train, prompt, and predict: A systematic survey of prompting methods in natural language processing. *ACM Computing Surveys*, 55(9):1–35.

Louise Richardson-Self. 2021. *Hate speech against women online: Concepts and countermeasures*. Rowman & Littlefield.

Julian Risch, Anke Stoll, Lena Wilms, and Michael Wiegand. 2021. Overview of the GermEval 2021 shared task on the identification of toxic, engaging, and fact-claiming comments. In *Proceedings of the GermEval 2021 Shared Task on the Identification of Toxic, Engaging, and Fact-Claiming Comments*, pages 1–12, Duesseldorf, Germany. Association for Computational Linguistics.

Indira Sen, Mattia Samory, Claudia Wagner, and Isabelle Augenstein. 2022. Counterfactually augmented data and unintended bias: The case of sexism and hate speech detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 4716–4726, Seattle, United States. Association for Computational Linguistics.

Hariram Veeramani, Surendrabikram Thapa, and Usman Naseem. 2024. MLInitiative@WILDRE7: Hybrid approaches with large language models for enhanced sentiment analysis in code-switched and code-mixed texts. In *Proceedings of the 7th Workshop on Indian Language Data: Resources and Evaluation*, pages 66–72, Torino, Italia. ELRA and ICCL.

Jingfeng Yang, Hongye Jin, Ruixiang Tang, Xiaotian Han, Qizhang Feng, Haoming Jiang, Shaochen Zhong, Bing Yin, and Xia Hu. 2024. Harnessing the power of llms in practice: A survey on chatgpt and beyond. *ACM Trans. Knowl. Discov. Data*, 18(6).

Wenxuan Zhou, Sheng Zhang, Hoifung Poon, and Muhao Chen. 2023. Context-faithful prompting for large language models. In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 14544–14556, Singapore. Association for Computational Linguistics.