# COMP 766 Project Report

**Vivek Verma (260907312)**
Université de Montréal
MILA
`vivek.verma3@mail.mcgill.ca`

## Abstract

For our final project we extend work on annotator bias (Geva et al., 2019), to two additional datasets - HateXplain, and Biasly - for Classification task. These datasets measure different types of biases: HateXplain is for detecting hate speech, and Biasly for detecting subtle misogyny. In this way, we see the applicability of the original paper to other datasets and other domains of Natural Language Understanding. We extend the original work to multiple annotators per sample and run three kinds of experiments for measuring annotator bias. We find that HateXplain shows annotator bias, whereas Biasly doesn't present much evidence of this bias. We hypothesize that this is because the annotators of Biasly dataset are expert annotators and annotate roughly equal number of samples. We justify the applicability of experiments for our Classification tasks where the annotators are labelers in comparison with the original datasets where annotators are creators. We also highlight limitations of the original paper in terms of validity, and that the recommendation to separate out test set annotators might not be feasible or realistic. [1]

## 1 Introduction

For our final project, we extend the work on *annotator bias* as presented in Geva et al. (2019). This paper was part of the reading list for our course. According to the paper, when models are trained on datasets that are created using crowdsourcing, it can affect their ability to generalize. It can also lead to an overestimation of model performance, as the model picks up language patterns that correlate with labels. Since we mention this paper throughout our work, we will call it "the original paper". Our goal from this work is to check the presence of annotator bias in more recent datasets and derive further insights. We also present any shortcomings and limitations that we discover during the experiment.

The original paper works on three Natural Language Understanding (NLU) datasets: MNLI (Williams et al., 2018), OpenBookQA (Mihaylov et al., 2018), and CommensenseQA (Talmor et al., 2019), thus covering Natural Language Inference (NLI) and Question Answering (QA) tasks. In our work we will look at two different, more recent datasets: HateXplain (Mathew et al., 2020), and Biasly (Sheppard et al., 2024). The chosen datasets are aimed at measuring or detecting different types of biases, and our work will be on binary and multiclass Classification.

A note here that in the project proposal we said we would also look at StereoSet (Nadeem et al., 2021). We realized that Stereoset is based on Next Sentence Prediction task, and the idea is to evaluate model performance without fine-tuning. We did not find any easy way to extend the work of the original paper which requires fine-tuning with annotator IDs and chose to focus on the other two datasets and on Classification tasks.

Geva et al. (2019) conduct three main experiments. First, they add annotator ID to the input features to establish that improves model performance. Second, they show that annotator information is captured by the models when models are able to recognize annotators that generate many examples. Finally, they test whether models generalize to annotators not seen at training time, by creating a different split for training and development set with disjoint annotators.

The main findings of the original paper are that model performance usually improves when training with annotator identifiers as features which means that models can pick up annotator patterns and recognize the most productive annotators. They also claim that models do not generalize to samples from unseen annotators and in this regard propose that test set should have samples from disjoint set

[1] Our code is available here

of annotators. They present this test-set split to be the main way of tackling annotator-bias.

Our main contributions with this work are as follows:

1) We justify and provide an extension of the original work to Classification tasks where annotators are labelers. 2) We provide an extension to all three experiments, to handle multiple annotators per sample. 3) We show through the third experiment, that the suggestion to have a test set with different annotators can lead to big variance in results and potentially misleading findings. 4) We highlight limitations from the original paper about poorly defined Systematized concept (Adcock and Collier, 2001) and lack of analysis of what it means for a dataset to be free from annotator bias.

## 2 Methodology

We describe below the datasets that we've chosen, the language model that we're going to use, and the metrics for reporting results.

### 2.1 Datasets

Both datasets have publicly available annotator information. We see statistics on the number of samples and number of annotators in Table 1. The distribution of annotations is shown in Figure 1. We see that top few annotators on HateXplain annotate a large number of samples and there is long tail of annotators that annotate few samples. For Biasly, the ten annotators each annotate roughly equal number of samples, so the graph is a straight line.

| Dataset | # Samples | # Annotators |
|---|---|---|
| HateXplain | 19,229 | 253 |
| Biasly | 9,935 | 10 |

Table 1: Statistics for datasets used in our experiments.

- **HateXplain**: This is a benchmark dataset on hate speech which was created to improve explainability of models in classifiying text as hate speech. The datasets contains posts from social media websites Twitter and Gab, and annotations cover three facets. First, the annotators classify each post as hate, offensive, or normal speech. Second, the annotators are asked to select the target communities mentioned in the post. Finally, the annotators highlight parts of the text that justifies their classification decision. For our experiments we will only look at the first facet of multi-class classification of a post as hate, offensive, or normal speech.

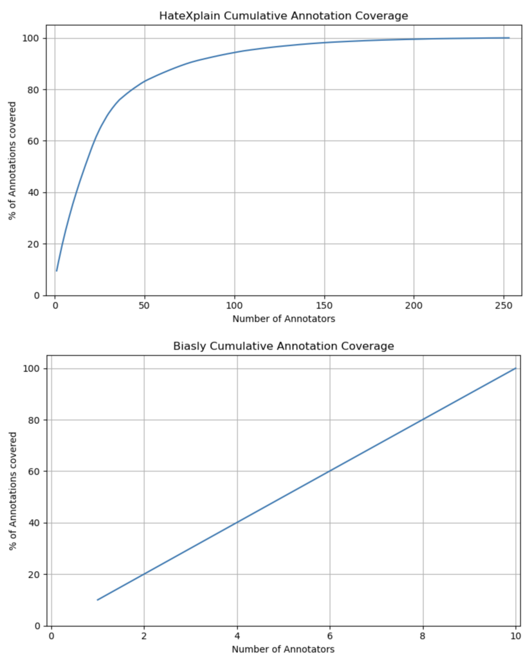

Figure 1: Proportion of annotations covered by number of annotators (sorted by number of annotations). Note that each sample has three annotations, so total number of annotations is three times the number of samples.

- **Biasly**: This dataset was created for subtle misogyny detection with misogyny defined as "Hatred of, dislike of, contempt for, or ingrained prejudice against women." It contains annotations of movie subtitles, capturing colloquial expressions of misogyny in North American film. Their rationale for using movie subtitle corpus over social media text was two fold: Firstly, social media contains overt forms of misogyny that can drown subtle forms of misogyny. Secondly, movie subtitles are similar to transcribed conversational speech. The data contains binary classification on the presence of misogynistic text, multi-label classification on types of misogynistic text, and a severity score. For our experiments we will only look at the binary classification of text is mysogynistic or not.

### 2.2 Language Model

Given the scope of the project, we will use BERT-base (Devlin et al., 2019) as our model of choice.

Both dataset papers report their performance on BERT, and BERT has either state-of-the-art (SOTA) or close to SOTA performance for all three of them.

For the experiments on HateXplain dataset we take the model parameters from their paper (Mathew et al., 2020). We use the same train:development:test split of 8:1:1, maximum input sequence is 128, batch size is 16, learning rate is 2e-5, and optimizer is Adam (Kingma and Ba, 2017), and training is done for 20 epochs.

For the experiments on Biasly dataset we look at the model parameters from the respective paper (Sheppard et al., 2024). We use the same train:development:test split of 8:1:1, maximum input sequence is 512, batch size is 32, learning rate is 2e-5, and optimizer is Adam (Kingma and Ba, 2017), and training is done for 3 epochs.

### 2.3 Experiments and Metrics

Our project works on the same three experiments as the original paper, for the chosen datasets.

We evaluate model performance on the first experiment using Accuracy and Macro F1 as our metrics and compare the numbers with and without annotator information. Both datasets use these metrics in their papers. For the second experiment we use F1 score as the metric to check for annotator recognition. It's the same metric that the original paper uses in their experiment. For the third experiment we again use Accuracy and Macro F1 and the check performance on different splits of test set.

## 3 Results

We now repeat the experiments in Geva et al. (2019) for measuring annotator bias, on our chosen datasets. We aim to answer the same three questions: 1) Do models perform better when exposed to the annotator ID? 2) Can models detect the annotators from their generated examples? 3) Do models generalize across annotators?

### 3.1 Experiment 1: The utility of annotator information

The goal of this experiment is to see if model performance increases when annotator information is provided. The way it's done in the original paper is that every sample $(x, y)$ created by annotator $z$, is replaced with the sample $((x, z), y)$, where $z$ is a textual unique annotator identifier, $x$ is the input sequence and $y$ is the gold label. But both our chosen datasets have three annotators per sample, so our data looks like:

$$((x, [(z_1, y_1), (z_2, y_2), (z_3, y_3)]), y)$$

where $z_i$'s are the annotators and $y_i$'s are the labels by these annotators. $y$ is the gold label which is the majority vote of $y_i$'s. Samples where majority vote does not exist, are discarded.

We handle this by splitting each of the samples into three different samples: $((x_1, z_1), y_1)$, $((x_2, z_2), y_2)$, $((x_2, z_2), y_2)$. Then on the validation and test sets we apply a majority vote on the output labels of the split samples to get the label for the original sample. Metrics are reported with respect to original un-split samples.

This choice of incorporating multiple annotators seemed the most natural to us. Each split sample is a complete sample that has the input $x$ and annotator preference. Other options we considered were adding all three annotators to the input - $((x, z_1, z_2, z3), y)$ but here we're losing information on individual preference and the results can potentially depend on order of the annotators. It's also not clear how a model would work annotator sequence is shuffled. We also thought about picking one annotator that agrees with the majority label but in this approach we throw away two-thirds of our data.

We show the results of the first experiment on the two datasets in Table 2. On the HateXplain dataset there is a big jump in performance when annotator ID is used as input. Overall accuracy improves from 0.675 to 0.726 and Macro F1 improves from 0.657 to 0.73. Because HateXplain dataset has three labels, majority vote might not exist on some samples where all three annotators choose a different label. There were 97 such cases in the test set, out of 1924. These are marked inaccurate for the accuracy calculations. The Macro F1 score is the average of the F1 score on the three labels.

On Biasly dataset, we see similar performance with and without annotator ID. With annotator ID, Accuracy has a small drop from 0.896 to 0.892, while Macro F1 has a small increase from 0.792 to 0.818. In this dataset there are only two labels, and hence amongst three annotators we don't have the problem of majority label not existing.

### 3.2 Experiment 2: Annotator recognition

The purpose of this experiment is to see if model can recognize annotators even without being exposed to annotator ID, only from the input. In the

| Dataset | Input Configuration | Accuracy | Macro F1 |
|---|---|---|---|
| HateXplain | Without Annotator ID | 0.675 | 0.657 |
| | With Annotator ID | 0.726 | 0.730 |
| Biasly | Without Annotator ID | 0.896 | 0.792 |
| | With Annotator ID | 0.892 | 0.818 |

Table 2: Performance comparison with and without annotator ID on HateXplain and Biasly datasets.

original paper they fine-tune BERT-base to predict annotator IDs from examples. The task is multi-class classification with 6 labels of the top-5 most productive annotators and an OTHER label for all other annotators. So, every example $(x, y)$ created by annotator $z$ is replaced with $(x, z)$ where $z$ now represents the label for the input $x$.

In our datasets, the input $x$ which is the sentence for classification is the same for all three annotators of that sentence. Hence, doing the same process would not make much sense as the annotators aren't the ones who created $x$. Conceptually we can extend this by adding annotator input which is the label for the sentence and then see if model can recognize annotators. We do this by converting the problem from

$$((x, [(z_1, y_1), (z_2, y_2), (z_3, y_3)]), y)$$

to $((x, y_i), z_i)$ for each of the three annotators. The new input being $x, y_i$ with the label being $z_i$. We don't expect the model performance for recognition to be good. If annotators have same labels for the input, which is true for at least 2 annotators in each sample, then the same input maps to different annotator ID outputs under this approach.

The results of the experiment for HateXplain dataset are in Table 3. The F1 scores are low for all the top-5 annotators, all below 0.1, with no clear trend. On Biasly, the model had F1 score of 0 for all the top 5 annotators. The OTHER label had a perfect recall score, which means the model predicted the annotator to be OTHER for each sample. Each annotator in this dataset had roughly the same amount of annotations of around 10% of the dataset for the 10 annotators.

### 3.3 Experiment 3: Generalization across annotators

In this experiment, the original paper wants to see whether fine-tuned models generalize to new annotators. The datasets are re-split to create new training, validation, and test sets so that the test

| Annotator | Percentage of Samples | F1 Score |
|---|---|---|
| A1 | 9.5 | 0.07 |
| A2 | 3.4 | 0.02 |
| A3 | 3.3 | 0.10 |
| A4 | 3.2 | 0.06 |
| A5 | 2.9 | 0.09 |

Table 3: For HateXplain, their top 5 annotators by size, their respective proportion of annotations in the dataset, and the F1 score of BERT model fine-tuned for annotator recognition.

set has disjoint annotators from the other two sets. There are two kinds of sub-experiments. The first kind is *single-annotator* split where the test set consists of all the samples from a single annotator. This is done for all the top-5 annotators separately. The second kind is *multi-annotator* split where a set of 5 annotators that produced low number of samples is taken to be in the test set.

To produce the splits for our datasets with multiple annotators per sample, we take all the samples with a particular annotator if we want that annotator to not be a part of the training set. This means that test set is comprised of these unseen annotators, but also has annotations from other annotators seen in the training set.

The original paper repeats the experiments multiple times to account for variance, but we do them only once for each split in consideration of resources and time. In addition to this, we only do them for the top-3 annotators in the single-annotator split experiment for both datasets. For HateXplain and the multi-annotator split experiment, we take set of five annotators that contribute to around 1% of the annotations. This is because the least productive annotators produce very few annotations for the test set to have reliable results. We do this three times. For Biasly, all the annotators contribute similar amount, and separating out a single annotator, moves 30% the data to the test set. Hence we cannot do the multi-annotator split experiments.

The results of the experiment for HateXplain are in Table 4. For both the single-annotator and multi-annotator experiments we see that sometimes performance degrades and sometimes performance improves on the split dataset offering no clear trend. We wanted to see if voting patterns for those annotators, in comparison to other annotators in the sample, is a factor in performance. For this purpose we show an Agreement percentage for each split. The number is the percentage of samples in the set where the label of the annotator agrees with the majority label of the sample.

On single-annotator experiments, we see that $A2$ has an extremely high $98\%$ agreement score, and the model performs better than baseline on this set. $A3$ has $78\%$ agreement score and model performs worse than baseline. Presumably demonstrating that the voting pattern is similar to that learned in the training set. $A1$ has $88\%$ agreement but the size of the test set is much larger. We can hypothesize that the proportion of the test set could also play a factor in performance. Having a test set that is $28\%$ of the size of the dataset, with a new annotator, could be difficult.

On multi-annotator experiments we see similar pattern. $S3$ has $68\%$ agreement and lower than baseline scores, wheareas $S1$ has $83\%$ agreement and higher than baseline scores.

We see single-annotator split performance on Biasly dataset in Table 5. The top-3 annotators all have split size of $30\%$. We see a small drop in accuracy compared to baseline for each of the three experiments and the Macro F1 score stays around the same. We notice that all three annotators have a high agreement score of above $90\%$. We can hypothesize that the small changes in performance compared to the baseline are due to high agreement scores.

## 4 Validity

The original paper talks about annotator-bias as follows: "In this paper, we continue recent efforts to understand biases that are introduced during the process of data creation. We investigate this form of bias, termed annotator bias." To analyze the validity of this, we can use the framework provided by Adcock and Collier (2001) which talks about measurement validity. They break it down into 4 levels - Background concept (level 1), Systemized concept (level 2), Indicators (level 3), and Scores (level 4) along with broad tasks to be performed to move between the levels. Background concept refers to the broad theoretical idea that a researcher wants to study, Systematized concept refers to a more concrete definition of the Background concept, Indicators are the measurable elements that operationalize the systematized concept, and Scores are the actual measurements for the Indicators.

Analyzing the original paper under this framework, we can say that the Background concept is about measuring bias from annotators where the authors want to quantify the impacts of annotators on datasets. The Systematized concept can be considered their definition of annotator bias in the previous paragraph. The Indictors are the three experiments and the Scores are the metrics. Where we see a fault of the original paper is in the Systematized concept. We feel the term annotator bias is not concretely defined and the idea is presented vaguely. From the definition, it is hard to tell if they mean deviation from ground truth, or culture differences of annotators, or consistent labeling patterns. Our idea of looking at agreement scores for the third experiment is to see deviation from majority vote, which can be considered a proxy for ground truth. In the first experiment, the model looks for patterns from annotators that can help improve performance. The definition of the term annotator bias should have been properly stated to avoid misunderstanding. However, we see no issues with the chosen Indicators and Scores.

## 5 Discussion and Conclusion

The original paper looks at annotators that are creators of the dataset. In our experiments the annotators are labelers of the dataset so we are looking at a slightly different problem that requires some justification. We feel that the approach is still valid, since for a classification task, the labels $y$ together with the text $x$ is what makes the dataset complete. For MNLI dataset, the annotator comes up with hypothesis and labels, for a given premise. The premise, hypothesis, and the label together is what makes a sample in the MNLI dataset, and similarly we can say that the text, and the annotator label is what makes a sample in HateXplain or Biasly dataset.

We've replicated all the experiments of the original paper, on two new datasets and for classification tasks. HateXplain had multi-class classification and Biasly had binary-classification. We extended

| | Single Annotator | | | | Multi Annotator | | |
|---|---|---|---|---|---|---|---|
| **Annotator** | **Accuracy** | **Macro F1** | **Agreement** | **Annotator** | **Accuracy** | **Macro F1** | **Agreement** |
| Baseline (10%) | 0.67 | 0.65 | - | Baseline(10%) | 0.67 | 0.65 | - |
| A1 (28%) | 0.60 | 0.57 | 88% | S1 (1%) | 0.79 | 0.72 | 83% |
| A2 (10%) | 0.74 | 0.79* | 98% | S2 (1%) | 0.72 | 0.64 | 78% |
| A3 (10%) | 0.58 | 0.59 | 78% | S3 (1%) | 0.59 | 0.59 | 68% |

Table 4: Single-annotator and Multi-annotator performance on the HateXplain dataset. For $A2$, the asterix on F1 score denotes that we took the weighted F1 score instead of Macro F1. This is because one of the labels had very few samples (17) and was distorting the overall F1 score much more than what was representative of performance. The percentage next to the annotator denotes the size of the test set. Baseline denotes the performance of the model on the original test set. The agreement score denotes the percentage of samples where the annotator label agrees with the majority label for the sample.

| Annotator | Accuracy | Macro F1 | Agreement (%) |
|---|---|---|---|
| Baseline (10%) | 0.90 | 0.79 | – |
| A1 (30%) | 0.88 | 0.80 | 95 |
| A2 (30%) | 0.88 | 0.79 | 90 |
| A3 (30%) | 0.88 | 0.78 | 92 |

Table 5: Single-annotator split performance on the Biasly dataset. All three top annotator splits make up 30% of the dataset. The agreement score denotes the percentage of samples where the annotator label agrees with the majority label for the sample.

the approach of the original paper to these datasets and provided a mechanism to deal with multiple annotators for a sample. For the first experiment, we saw that on HateXplain there was a clear increase in performance whereas on Biasly there was no clear change in performance supply annotator information. Conceptually, additional information should lead to better performance. So, Biasly does not demonstrate annotator bias in this experiment whereas HateXplain does.

We provided a way to extend the second experiment for annotator recognition but it is not clear whether this approach gives us much information. The classification tasks have few labels and the same inputs map to different annotators making correct classification difficult for models. F1 scores of annotator recognition on HateXplain were low and all below 0.1. On Biasly the model could not recognize the any of the top annotators and had F1 score of 0 for all of them.

For the third experiment, separating out annotators for test set meant taking all samples where a particular annotator is amongst the set of annotators for that sample. On single-annotator split this would mean that the test set still has annotations from two other annotators per sample, but at least a third of the annotations are from a new annotator. For HateXplain, we saw performance on the test set vary across different annotators with no clear trend and our hypothesis is that this is a factor of the agreement score and test set size. Looking at these results, we don't agree with the recommendations of the original paper to have separate test set annotators. Annotators that are separated out for the test set might have low agreement with other annotators for various reasons and this might not be indicative of generalization capabilities of the model. Seeing that depending on the split, we can end up with higher or lower performance, with no clear trend, this recommendation is difficult to incorporate. The variance in results was similar in the original paper. On Biasly, model performance on the split datasets, does not change much compared to baseline.

Naturally, looking at all our experiment results, we have to ask here why is it that we see a stark difference in behaviour between HateXplain and Biasly datasets? Biasly dataset was annotated by 10 expert annotators. Each of them with roughly the same number of annotations and individually high agreement scores with the majority label. This makes the dataset more homogenous in some sense. We saw no increase in performance with annotator information, no annotator recognition, and not much reduction in performance when using separate annotators on the test set. We can only conjecture that the annotation properties of this set that we've mentioned, play a key role in the dataset not showing annotator-bias.

In contrast, HateXplain has 253 annotators and their share of annotations ranging for 28% to a handful of samples. The annotators are from MTurk and not expert annotators. We also see a wide difference in agreement scores. These differences with Biasly are perhaps the reasons for the differences in performance.

It is unclear whether the lack of annotator-bias on the Biasly dataset makes it a better dataset and it is not clear what the impacts are of having an annotator-bias free dataset. To answer this question concretely would require different versions of the same dataset and some understanding of the gold standard, to see whether annotator-bias free dataset is closer to this gold standard. This is theoretically an expensive exercise and might not be possible to do. This is a limitation from the original paper. We see here some properties of the Biasly dataset that might have contributed to it being free from annotator-bias, expert annotators being one of them. But getting expert annotators is not always possible. Perhaps it could be possible to strive for roughly similar number of annotations from different annotators which is another possible factor for presence of annotator-bias.

## 6 Limitations

Further work on this topic has revealed other insights. Annotator patterns can occur because of demographic characteristics of annotators (Al Kuwatly et al., 2020). In detection of toxic language, bias can come from annotator beliefs and identities (Sap et al., 2022). Bias can also originate from annotation instructions (Parmar et al., 2023) but since we're looking at only classification tasks instructions should not play much role as compared to other tasks where annotators generate complete sentences. In our work we did not look at any of these aspects due to lack of detailed information about the annotators in the dataset, and because of the scope of the project.

## References

Robert Adcock and David Collier. 2001. Measurement validity: A shared standard for qualitative and quantitative research. *American Political Science Review*, 95(3):529–546.

Hala Al Kuwatly, Maximilian Wich, and Georg Groh. 2020. Identifying and measuring annotator bias based on annotators' demographic characteristics. In *Proceedings of the Fourth Workshop on Online Abuse and Harms*, pages 184–190, Online. 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.

Mor Geva, Yoav Goldberg, and Jonathan Berant. 2019. Are we modeling the task or the annotator? an investigation of annotator bias in natural language understanding datasets. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 1161–1166, Hong Kong, China. Association for Computational Linguistics.

Diederik P. Kingma and Jimmy Ba. 2017. Adam: A method for stochastic optimization. *Preprint*, arXiv:1412.6980.

Binny Mathew, Punyajoy Saha, Seid Muhie Yimam, Chris Biemann, Pawan Goyal, and Animesh Mukherjee. 2020. Hatexplain: A benchmark dataset for explainable hate speech detection. *CoRR*, abs/2012.10289.

Todor Mihaylov, Peter Clark, Tushar Khot, and Ashish Sabharwal. 2018. Can a suit of armor conduct electricity? a new dataset for open book question answering. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 2381–2391, Brussels, Belgium. Association for Computational Linguistics.

Moin Nadeem, Anna Bethke, and Siva Reddy. 2021. StereoSet: Measuring stereotypical bias in pretrained language models. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 5356–5371, Online. Association for Computational Linguistics.

Mihir Parmar, Swaroop Mishra, Mor Geva, and Chitta Baral. 2023. Don't blame the annotator: Bias already starts in the annotation instructions. In *Proceedings of the 17th Conference of the European Chapter of the Association for Computational Linguistics*, pages 1779–1789, Dubrovnik, Croatia. Association for Computational Linguistics.

Maarten Sap, Swabha Swayamdipta, Laura Vianna, Xuhui Zhou, Yejin Choi, and Noah A. Smith. 2022. Annotators with attitudes: How annotator beliefs and identities bias toxic language detection. In *Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5884–5906, Seattle, United States. Association for Computational Linguistics.

Brooklyn Sheppard, Anna Richter, Allison Cohen, Elizabeth Smith, Tamara Kneese, Carolyne Pelletier, Ioana Baldini, and Yue Dong. 2024. Biasly: An expert-annotated dataset for subtle misogyny detection and mitigation. In *Findings of the Association for Computational Linguistics: ACL 2024*, pages 427–452, Bangkok, Thailand. Association for Computational Linguistics.

Alon Talmor, Jonathan Herzig, Nicholas Lourie, and Jonathan Berant. 2019. CommonsenseQA: A question answering challenge targeting commonsense knowledge. 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 4149–4158, Minneapolis, Minnesota. Association for Computational Linguistics.

Adina Williams, Nikita Nangia, and Samuel Bowman. 2018. A broad-coverage challenge corpus for sentence understanding through inference. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 1112–1122, New Orleans, Louisiana. Association for Computational Linguistics.