

CS60075: Project Report

Group-2

Shubhraneel Pal [19CS30054]

Aryan Mehta [19CS30006]

Debanjan Saha [19CS30014]

Deep Majumder [19CS30015]

Krishna Yadav [19EE10035]

Problem Description

We were assigned the task of **Detection of Online Sexism**

Sexism is defined as any abuse or negative sentiment that is directed towards women based on their gender or based on their gender combined with one or more other identity attributes (e.g. Black women, Muslim women, Trans women).

Sexism is a growing problem online. It can inflict harm on women who are targeted, make online spaces inaccessible and unwelcoming, and perpetuate social asymmetries and injustices. Automated tools are now widely deployed to find, and assess sexist content at scale but most only give classifications for generic, high-level categories, with no further explanation. Flagging what is sexist content and also explaining why it is sexist improves interpretability, trust and understanding of the decisions that automated tools use, empowering both users and moderators.

The task contains three hierarchical subtasks:

- **TASK A** - Binary Sexism Detection: a two-class (or binary) classification where systems have to predict whether a post is sexist or not sexist.
- **TASK B** - Category of Sexism: for posts which are sexist, a four-class classification where systems have to predict one of four categories: (1) threats, (2) derogation, (3) animosity, (4) prejudiced discussion.
- **TASK C** - Fine-grained Vector of Sexism: for posts which are sexist, an 11-class classification where systems have to predict one of 11 fine-grained vectors.

Dataset

Several posts from Gab and Reddit were collected and shared with us among three groups who are working on this project. We were given a chunk of this dataset split randomly among the three groups.

Some statistics of the dataset are as follows:

Dataset Type	Number of Posts
Train	8000
Test	2000

Each of these datasets contains four columns:

1. **text**: This field contains the contents of the post
2. **label_sexist**: This field contains the ground truth value whether the posted content is sexist or not
3. **label_category**: This field contains the type of sexism for a sexist content
4. **label_vector**: This field contains the fine-grained vector of sexism for a sexist content.

Exploratory Data Analysis

Upon basic data analysis on the **training** dataset, we found out the following statistics about our training set for each classification task.

Sexist Content Statistics

S/L	Type of Post	Number of Posts
1	Sexist	1997
2	Not Sexist	6003

Types of Sexism

Four types of sexism were present in the dataset - (1) threats, (2) derogation, (3) animosity, (4) prejudiced discussion. Out of the 1997 sexist posts, the statistics for each type of sexist post is given as follows:

S/L	Type of Sexism	Number of Posts
-----	----------------	-----------------

1	descriptive attacks	924
2	animosity	685
3	threats, plans to harm and incitement	198
4	prejudiced discussions	190

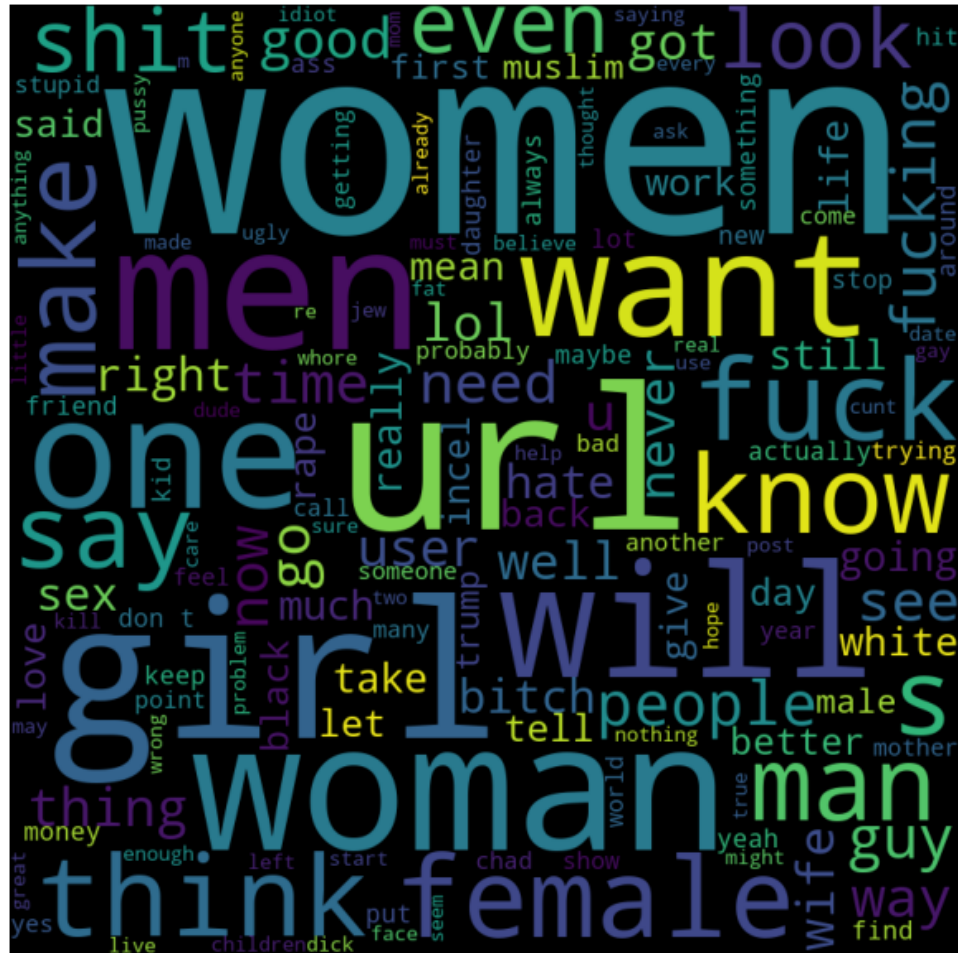
Fine-grained Vector of sexism

11 types of fine-grained types of sexism were present in the dataset. Out of the 1997 sexist posts, the statistics for each type of fine-grained sexist post is given as follows:

S/L	Fine-grained vector of sexism	Number of Posts
1	derogation	410
2	aggressive and emotive attacks	386
3	casual use of gendered slurs, profanities, and insults	376
4	immutable gender differences and gender stereotypes	253
5	incitement and encouragement of harm	160
6	supporting systemic discrimination against women as a group	146
7	dehumanising attacks & overt sexual objectification	128
8	supporting mistreatment of individual women	44
9	threats of harm	38
10	backhanded gendered compliments	34
11	condescending explanations or unwelcome advice	22

Text analysis

Here is a word cloud formed from the text of the posts present in the training dataset. As we can see, several vulgar words pop up due to the sexist content present in the text of the dataset.



Model Description and Configuration

We use a pretrained BERT model and finetune it by adding a classification head to the output of the CLS token. The classification head is a single linear layer from 768 to the number of classes followed by a softmax activation. We initially tried using a model that directly classifies the number of classes at each of the 3 levels. That is, for the first task the classification head goes from 768 to 2, for the second level task from 768 to 5 and for the final level task from 768 to 12. Each of these models were trained separately starting from the official pretrained BERT checkpoint and keeping the parameters of the BERT architecture trainable.

Another thing we tried was using the information learned for predicting a higher level class to predict the lower level class. For this we started training the 5-class classifier from the trained 2-class classifier. That is, the weights of BERT in the 2 class classifier are copied initially in the weights of BERT in the 5-class classifier and then the training is done. Similar is done for the 12-class classifier from the 5-class classifier.

We then perform the same experiments with RoBERTa instead of BERT.

Model Link:

Drive Link: https://drive.google.com/drive/u/0/folders/1O_yXIjarYi7shPMzhmpk4F4ieTV83Lzz

The names of the models and their descriptions are given below

2class_model.pth	BERT classifier for sexist and non_sexist
5class_model.pth	BERT classifier for second level of classification
12class_model.pth	BERT classifier for third level of classification
5class_finetuned_model.pth	BERT classifier for second level of classification finetuned from 2class_model.pth
12class_finetuned_model.pth	BERT classifier for third level of classification finetuned from 5class_model.pth
2class_model_roberta.pth	RoBERTa classifier for sexist and non_sexist
5class_model_roberta.pth	RoBERTa classifier for second level of classification
12class_model_roberta.pth	RoBERTa classifier for third level of classification
12class_finetuned_model_roberta.pth	RoBERTa classifier for third level of classification finetuned from 5class_model_roberta.pth

Results

Model	Accuracy (%)	F1 (%)
2class_model.pth	77.4	71.4
5class_model.pth	79.4	74.0
12class_model.pth	75.8	69.6
5class_finetuned_model.pth	77.3	71.3
12class_finetuned_model.pth	78.4	72.6
2class_model_roberta.pth	84.3	79.8
5class_model_roberta.pth	79.7	74.3
12class_model_roberta.pth	78.5	72.8
12class_finetuned_model_roberta.pth	77.6	71.8

2-Class testing


```
[ ] test_dataset = Dataset(text, label_sexist)
```

```
[ ] model = BertClassifier(num_classes=2)
model.load_state_dict(torch.load("/content/drive/MyDrive/NLP Project/models/2class_model_roberta.pth"))
```

Downloading: 100%  436M/436M [00:07<00:00, 56.7MB/s]

Some weights of the model checkpoint at bert-base-cased were not used when initializing BertModel: ['cls.predictions.bias']
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or with another architecture (e.g. initializing a BertForSequenceClassification model from a BertForPreTraining model).
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be initialized with the same weights.
<All keys matched successfully>

```
[ ] evaluate(model, test_dataset)
```

100%  | 1000/1000 [01:12<00:00, 13.81it/s] Test Accuracy: 0.843
Test F1-score: 0.798

5-Class Testing

```
[ ] test_dataset = Dataset(text, label_category)
```

```
▶ model = BertClassifier(num_classes=5)
  model.load_state_dict(torch.load("/content/drive/MyDrive/NLP Project/models/5class_model_roberta.pth"))
```

Some weights of the model checkpoint at bert-base-cased were not used when initializing BertModel: ['cls.predicti
- This IS expected if you are initializing BertModel from the checkpoint of a model trained on another task or wi
- This IS NOT expected if you are initializing BertModel from the checkpoint of a model that you expect to be exa
<All keys matched successfully>

```
[ ] evaluate(model_5class_roberta, test_dataset)
```

```
100%|██████████| 1000/1000 [01:12<00:00, 13.82it/s]Test Accuracy: 0.797
Test F1-score: 0.743
```

12-Class classification

```
▶ test_dataset = Dataset(text, label_vector)
```

```
[ ] model = BertClassifier(num_classes=12)
  model.load_state_dict(torch.load("/content/drive/MyDrive/NLP Project/models/12class_model_roberta.pth"))
```

Some weights of the model checkpoint at roberta-base were not used when initializing RobertaModel: ['lm_head.dense.wei
- This IS expected if you are initializing RobertaModel from the checkpoint of a model trained on another task or with
- This IS NOT expected if you are initializing RobertaModel from the checkpoint of a model that you expect to be exact
<All keys matched successfully>

```
[ ] evaluate(model, test_dataset, num_classes=12)
```

```
100%|██████████| 1000/1000 [01:03<00:00, 15.69it/s]Test Accuracy: 0.785
Test F1-score: 0.728
```

Inference

The performance improves a little bit when doing pre training on a previous level. But it is not that significant. To get a larger increase due to pretraining, we need a larger 2 class dataset.

The performance is seen to improve a lot with RoBERTa as RoBERTa is robust due to bigger training set and dynamic masking in the pretraining.