Online Sexism Detection

1. Introduction

Sexism is increasingly prevalent in online environments, posing risks to targeted women, creating barriers to inclusivity, and reinforcing societal inequalities. While automated tools are now commonly used to detect and evaluate sexist content on a large scale, many provide only broad classifications without detailed explanations. By not only identifying sexist content but also providing clear explanations for why it is deemed as such, these tools can enhance transparency, trust, and comprehension of automated decisions, benefiting both users and moderators alike. In our project, we mainly did 2 tasks. One is sexism detection and then sexism category detection. For the first task, it was a binary classification, the class labels that we tried to predict were ‘sexist’ and ‘not sexist’. For the later task, it was a multiclass classification, we tried to predict ‘threats, plans to harm and incitement’, ‘prejudiced discussions’. ‘animosity’, and ‘derogation’.

2. Dataset Description

The full dataset contains lots of data and files, but since our target was to use only supervised learning, our options became limited to one file. It contained 14000 instances of data. There were few labels in the data, but our task was limited to only two task. The first task is sexism detection and the second one is what type of sexism it was. The data was quite unbalanced as shown in Figures 1 & 2. Here, figure 1 represents data for task 1 and Figure 2 shows the data for task 2. Figure 3 & 4 contain word cloud for task 1 and task 2 respectively.

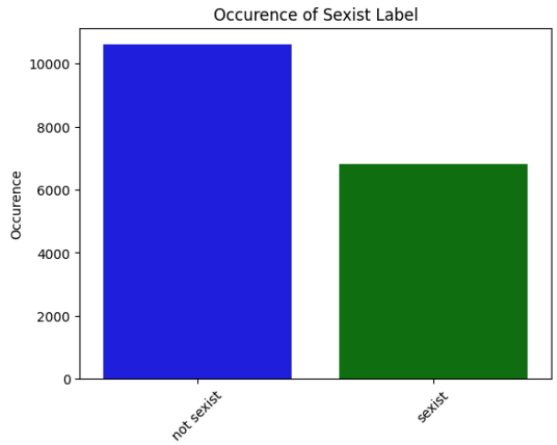

Figure 1: Data Distribution for Sexism Detection, Task 1

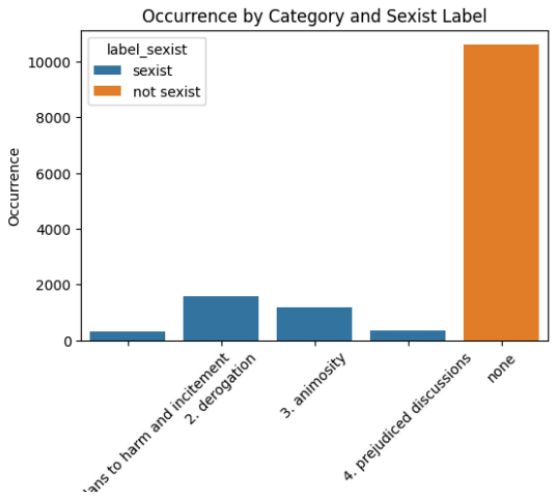

Figure 2: Data Distribution for Category of Sexism Detection, Task 2

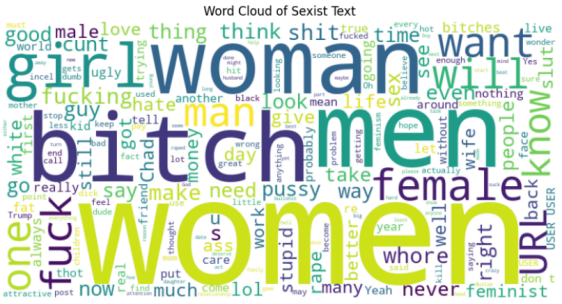

Figure 3: Word Cloud for Task 1

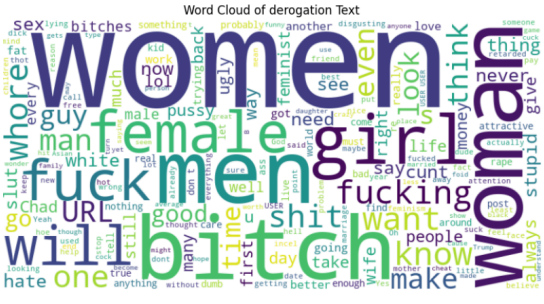

Figure 4: Word Cloud for Task 2

2.1 Data Augmentation

The dataset was heavily imbalanced for both tasks. For this, we used data augmentation. The data was augmented using *word2vec*. The library we used for this is ***nlpaug*.** It inspected each sentence and tried to find the most common or similar it could find in the vector representation it has. It then replaced those words with the most common word it found. After augmenting the data, the data for both tasks became balanced. The total instances for task 1 was after augmentation was 17398 and before that it was 14000. For task 2, after augmentation, we had 6395 instances. Before augmentation, it was 3398.

2.2 Data Preprocessing

In the course of this NLP task, several pivotal preprocessing steps were undertaken. Initially, the text underwent purification, involving the removal of special characters, standardization to lowercase, and elimination of stopwords to enhance semantic clarity. Subsequently, tokenization was employed using the text_to_sequences method, facilitating the conversion of textual data into numerical sequences. To ensure uniformity in input dimensions, padding techniques were applied.

3. Models & Embedding

3.1 Fully Connected Neural Network (FCNN)

Fully Connected Neural Networks (Fig 5) excel at mapping complex decisions by leveraging interconnected layers of neurons. Each neuron receives inputs from every neuron in the previous layer, enabling intricate pattern recognition and decision-making.

3.2 Recurrent Neural Networks (RNN)

Recurrent Neural Networks (Fig 6) process sequences of data by retaining memory of previous inputs, making them adept at tasks requiring context, like language translation or sentiment analysis. Their sequential processing allows for understanding temporal dependencies, facilitating complex decision-making.

3.3 Recurrent Neural Networks (RNN)

Long Short-Term Memory (Fig 7) networks, a type of recurrent neural network, excel at capturing long-range dependencies in sequential data, making them ideal for tasks requiring context retention and complex decision-making.

3.4 Gated Recurrent Units (GRU)

Gated Recurrent Units (Fig 8), akin to LSTM, efficiently capture long-range dependencies in sequential data. Their simplified architecture aids in memory retention, facilitating context-based decision-making in tasks like language modeling.

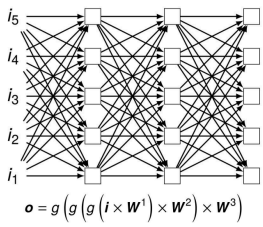

Fig 5: FCNN

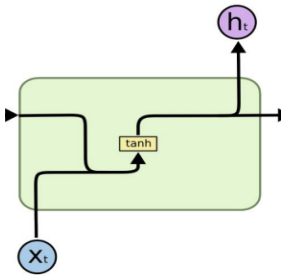

Fig 6: RNN

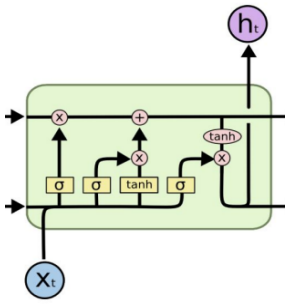

Fig 7: LSTM

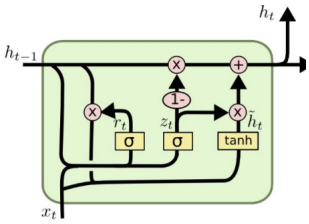

Fig 8: GRU

3.5 GloVe (Global Vectors for Word Representation)

GloVe embeds words in a continuous vector space based on their co-occurrence statistics in a corpus, facilitating semantic understanding. Incorporated as an embedding layer in various NLP models, including RNNs, LSTMs, and GRUs, GloVe enhances their ability to capture contextual information and make informed decisions.

3.5 Result Analysis

The final performance of the models is presented in a tabular format in Table 1 and Table 2 for the two tasks respectively. For task 1, we can see that GRU outperformed LSTM by 1%. For task 1, as we can see from the table, all the models performed similarly apart from FCNN. FCNN is poor at capturing sequences. Our data was sequential so FCNN performed a bit poorly but it's still acceptable. This is because it earned a fair precision, recall, and f1-score. Also, a thing to note that LSTM took the most time to run, but GRU outperformed it and also took less amount of time to run. So we would prefer GRU over LSTM in this case.

	Accuracy	Precision	Recall	F1-Score	Model Training Time (seconds)
Fully Connected Neural Network	0.73	0.73	0.73	0.73	180
Recurrent Neural Network	0.81	0.81	0.81	0.81	375
Long Short-Term Memory	0.82	0.82	0.82	0.82	1200
Gated recurrent unit	**0.83**	**0.83**	**0.83**	**0.83**	900

Table 1: Task 1 Performance Table

For the second task, we can see that LSTM performed best. As we know it's great at remembering important information. RNN faced a slight drop in performance. As this is a multiclass classification, we can see there is a significant performance drop in FCNN. This table shows the capability of FCNN really falls when it comes to sequential data. In comparison to task 1, the LSTM, RNN, GRU performed quite well given this was a 4-class classification.

	Accuracy	Precision	Recall	F1-Score	Model Training Time (seconds)
Fully Connected Neural Network	0.55	0.56	0.55	0.55	75
Recurrent Neural Network	0.68	0.67	0.68	0.67	90
Long Short-Term Memory	**0.73**	**0.72**	**0.73**	**0.72**	240
Gated recurrent unit	0.71	0.71	0.71	0.71	210

Table 2: Task 2 Performance Table

5 Limitations

Although we achieved quite good acceptable results for both tasks, we could've achieved better performance if we somehow utilized the unlabeled data. Also, I tried to apply Bidirectional Encoder Representations from Transformers (BERT) for this task. However, I was facing some coding errors for task 1. Although I managed to ran it on task 2, but it was giving very poor performance of 25% accuracy. So a future task could be to apply BERT for this task. Another important thing I also tried to implement was using Explainable AI (XAI) using LIME & SHAP. But due to some library mismatch error couldn't present the results. We need this XAI so that we can trust the model and verify if the model is actually doing what it should. It will help us understand that our if model is making predictions based on credible features.