

**CSE440: Natural Language Processing II**

**PROJECT REPORT**

**Title:** Explainable Detection of Online Sexism (EDOS)

Section: 01

Group ID: 18

**Group Members:**

1) Ashfaq Ahmad Saad (**21301665**)

3) Asif Ahnaf Chowdhury (**21301510**)

4) A N M Jubair Tanvir (**21301524**)

**Table of contents:**

| **Serial** | **Content** |
| --- | --- |
| 01. | Introduction |
| 02. | Dataset |
| 03. | Model |
| 04. | Result Analysis |

**Introduction:** Sexism online is becoming a big problem. It harms women who are targeted, making online spaces unwelcoming and hard to access. This unfair treatment also keeps unfair social differences alive. Nowadays, automated tools are used a lot to find and judge sexist content in large amounts. But often, these tools only give basic labels without explaining why something is sexist. This makes it hard to understand and trust their decisions. By pointing out what is sexist and explaining why, we can make these tools more understandable and trustworthy for both users and moderators.

Our project aims to build English-language models for detecting sexism online. We want to make these models more accurate and easier to understand, with detailed categories for sexist content from places like Gab and Reddit.

We have two main tasks in our project:

Task A: Binary Sexism Detection. Here, the goal is to decide if a post is sexist or not sexist, putting it into one of two groups.

Task B: Category of Sexism. For posts that are sexist, we want to classify them into four different categories: threats, derogation (putting someone down), animosity (hostility), and prejudiced discussions.

This report will explain how we did our work, what we found, and what it means. Our hope is that by making these models better at spotting and understanding sexism online, we can make the internet a safer and fairer place for everyone.

**Dataset:** The dataset complimented both binary classification and multiclass classification. Important columns of the dataset were, text, label\_sexist, and label\_category. The label\_sexist column was used for binary classification and the label\_category was used for multiclass classification.  
Total datasample: 14000

TaskA downsampled to: 3398 each

TaskB downsampled to: 320 each category

Cleaned the data.

**MODEL**

1. **LSTM :**

**Model Architecture(taskA):**

**Embedding Layer:**

* Input dimension: vocab\_size = 500
* Output dimension: embedding\_dim = 16
* Input length: max\_len = 50

**Spatial Dropout Layer:**

* Dropout rate: drop\_lstm = 0.2

**LSTM Layer:**

* Number of units: n\_lstm = 128
* Return sequences: False (returns only the last output)

**Dropout Layer:**

* Dropout rate: drop\_lstm = 0.2

**Dense (Output) Layer:**

* Units: 1 (since it's a binary classification problem)
* Activation: Sigmoid

**Hyperparameters:**

* max\_len: 50 (maximum sequence length after padding/truncating)
* trunc\_type: 'post' (truncation type after the sequence)
* padding\_type: 'post' (padding type)
* oov\_tok: '<OOV>' (out of vocabulary token)
* vocab\_size: 500 (maximum number of words to keep based on frequency)
* n\_lstm: 128 (number of LSTM units)
* drop\_lstm: 0.2 (dropout rate for LSTM and Spatial Dropout layers)
* embedding\_dim: 16 (dimension of the embedding space)

**Training Details:**

* Loss Function: Binary Crossentropy
* Optimizer: Adam
* Metrics: Accuracy
* Early Stopping: Monitored on validation loss with patience of 2 epochs
* Number of Epochs: Maximum of 30 epochs or until early stopping criterion is met
* Training Data: training\_padded and y\_train
* Validation Data: testing\_padded and y\_test

**Learning Curve:**

The learning curve would show the trend of training and validation loss and accuracy over epochs. This curve can help in understanding if the model is overfitting or underfitting.

**Number of Layers:**

The model consists of 5 layers: Embedding, Spatial Dropout, LSTM, Dropout, and **Dense layers.**

**Number of Units:**

The LSTM layer has 128 units. The Dense layer (output layer) has 1 unit for binary classification.

1. **Dense Model:**

**Model Architecture:**

Embedding Layer:

Input dimension: vocab\_size = 500

Output dimension: embedding\_dim = 16

Input length: max\_len (not explicitly defined in the provided code)

**GlobalAveragePooling1D Layer:**

No parameters to define

**Dense Layer 1**:

Units: n\_dense = 24

Activation function: ReLU

**Dropout Layer 1:**

Dropout rate: drop\_value = 0.2

**Dense Layer 2 (Output Layer):**

Units: 1

Activation function: Sigmoid

**Hyperparameters:**

vocab\_size: 500 (maximum number of words to keep based on frequency)

embedding\_dim: 16 (dimension of the embedding space)

drop\_value: 0.2 (dropout rate for the Dropout layer)

n\_dense: 24 (number of units in the first Dense layer)

**Number of Layers:**

The model consists of 5 layers: Embedding, GlobalAveragePooling1D, Dense (with ReLU activation), Dropout, and Dense (output layer with Sigmoid activation).

**Number of Units:**

The first Dense layer has 24 units. The Dense layer (output layer) has 1 unit for binary classification.

**Result Analysis:**

For task A Dense model –

Precision measures the accuracy of positive predictions:

Precision for class 0 (label 0) is 0.65, indicating that 65% of instances predicted as class 0 are actually class 0.

Precision for class 1 (label 1) is 0.81, meaning that 81% of instances predicted as class 1 are actually class 1.

Recall measures the ability of the classifier to find all positive instances:

Recall for class 0 (label 0) is 0.87, indicating that 87% of all actual class 0 instances are correctly identified.

Recall for class 1 (label 1) is 0.55, meaning that 55% of all actual class 1 instances are correctly identified.

F1-score is the harmonic mean of precision and recall, providing a balance between them:

F1-score for class 0 (label 0) is 0.74.

F1-score for class 1 (label 1) is 0.65.

Accuracy is the overall correctness of predictions:

The model achieves an accuracy of 0.70, correctly predicting the class for 70% of instances in the test set.

Macro Avg represents the average of precision, recall, and F1-score across all classes without considering class imbalance:

Macro average precision is 0.73, recall is 0.71, and F1-score is 0.70.

Weighted Avg is the average of precision, recall, and F1-score with weights proportional to the number of true instances for each class:

Weighted average precision is 0.73, recall is 0.70, and F1-score is 0.70.

In summary, the model demonstrates reasonable performance in terms of precision, recall, and F1-score, with a slightly better performance on class 0 compared to class 1.