Project Report

Course Title: CSE 440 (Natural Language Processing II)

Submitted to:

Dr. Farig Yousuf Sadeque Sir

Submitted by

Name	ID
Md. Ahnaf Farhan	22241056
Munawar Mahtab Ansary	23341112
Avishek Paul	21301171
Aniqa Ibnat Jisa	20201136

Introduction

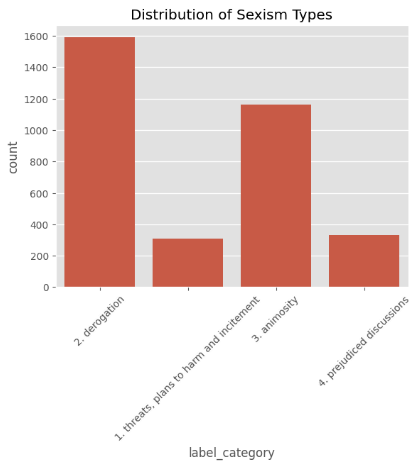
This project is focused on utilizing machine learning to classify instances of sexism sourced from the Explainable Detection of Online Sexism (EDOS) dataset, acquired from the Codalab competition SemEval 2023. Our objective is to construct a network model capable of comprehending and categorizing instances of sexism based on their textual content. Through this, we aim to demonstrate the capacity of machines to interpret language and identify instances of sexism, which has become increasingly crucial in addressing online toxicity and promoting a safer digital environment. We opted for a shallow recurrent network (RNN) due to its effectiveness in handling sequential data such as text. To augment its capabilities, we incorporated gated recurrence mechanisms by integrating a bidirectional LSTM layer. The insights and outcomes derived from this project have potential applications in various domains, including social media moderation tools and online community management systems. By enabling automated detection of sexism in online content, our work contributes to fostering more inclusive and respectful online interactions, thereby addressing important societal challenges in the digital age. The project consists of two primary objectives: Task A focuses on detecting instances of sexism within the text, while Task B involves categorizing the identified instances of sexism into specific groups.

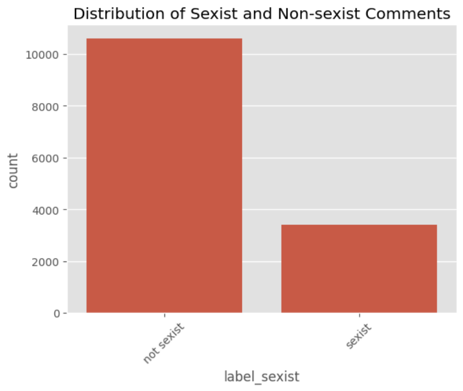
Dataset

The dataset has these columns:

- rewire_id: A unique identifier for each text entry.
- text: The actual text data containing statements or sentences.
- label_sexist: Indicates whether the text is considered sexist or not, with possible values being "sexist" or "not sexist".
- label_category: Describes what type of sexism(4type of sexism) and None for not_sexist
- label_vector: This column potentially represents another type of label or feature vector associated with the text. However, in this snippet, all entries are labeled as "none", suggesting that this column might not be used or populated in this subset of the data.

We have to work with task text and label_sexist for task A and label_category and for task B we have to work with text and label_category. The data is highly unbalanced for task A. For task B the data is also unbalanced but not as grave as task A . But task B has low amount of data.





Methodology

Dataset preparation:

- **Single Characters:**
  - Removing single characters surrounded by whitespace, such as 'a' or 'b'.
- **Non-Alphabetic Characters:**
  - Eliminating all characters that are not alphabetic letters, including numbers and special characters.
- **HTML Tags:**
  - Striping out any HTML tags present in the text, ensuring only the textual content remains.
- **Punctuation:**
  - Removing all punctuation marks from the text, such as commas, periods, and quotation marks.
- **Stopwords:**
  - Filtering out common stopwords, such as "the", "and", and "is", which do not carry significant meaning.

Text vectorization:

- Each review was converted into a numerical format using GloVe (Global Vectors for Word Representation) embeddings, which represent words as 100-dimensional vectors.

Data splitting:

- The dataset was divided into two parts: 80% for training the model and 20% for testing its performance for both tasks.

Model Building (Implementation of Bi-directional LSTM:

For task A

- **Input Layer (Embedding):**
  - Input shape: (256,)
  - Embedding layer with vocabulary size equal to len(word_index) + 1 and embedding dimension of 100.
  - The embedding matrix is initialized with pre-trained word embeddings provided by embedding_matrix.
  - This layer is trainable, allowing the model to fine-tune the embeddings during training.
- **Hidden Layer (Bidirectional LSTM):**
  - Bidirectional Long Short-Term Memory (LSTM) layer with 16 units.
  - Bidirectional LSTM processes the input sequence in both forward and backward directions, capturing dependencies in both directions.
- **Dropout Layer:**
  - Dropout layer with a dropout rate of 0.2.
  - Dropout is applied to regularize the model and prevent overfitting by randomly dropping 20% of the input units during training.
- **Hidden Layers (Dense):**
  - Two dense layers with 8 and 4 units, respectively, using ReLU activation functions.
  - Dropout layers with a dropout rate of 0.2 are applied after each dense layer for regularization.
- **Output Layer (Dense):**
  - Dense layer with 1 unit and sigmoid activation function.
- **Model Compilation:**
  - Loss function: Binary Crossentropy
  - Optimizer: Adam
  - Metrics: Accuracy, Precision, Recall

For task B

- **Input Layer (Embedding):**
  - Input shape: (256,)
  - Embedding layer with vocabulary size equal to len(word_index) + 1 and embedding dimension of 100.
  - The embedding matrix is initialized with pre-trained word embeddings provided by embedding_matrix.
  - This layer is trainable, allowing the model to fine-tune the embeddings during training.

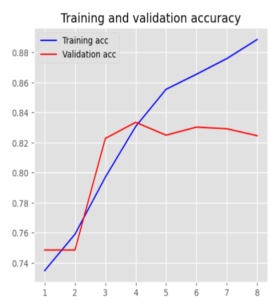
- **Hidden Layers (Bidirectional LSTM):**
  - First Bidirectional Long Short-Term Memory (LSTM) layer with 128 units and return sequences set to True.
  - Dropout layer with a dropout rate of 0.4 is applied after this layer for regularization.
  - Second Bidirectional LSTM layer with 64 units.
  - Dropout layer with a dropout rate of 0.4 is applied after this layer for regularization.
- **Hidden Layers (Dense):**
  - Dense layer with 8 units and ReLU activation function.
  - ReLU (Rectified Linear Unit) introduces non-linearity to the model.
  - Dropout layer with a dropout rate of 0.4 is applied after this layer for regularization.
- **Output Layer (Dense):**
  - Dense layer with 1 unit and sigmoid activation function.
- **Model Compilation:**
  - Loss function: Categorical Crossentropy
  - Optimizer: Adam
  - Metrics: Accuracy, Precision, Recall

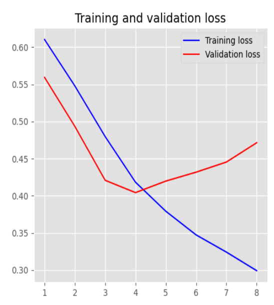
For task A we are using epoch 20 and batch size 25

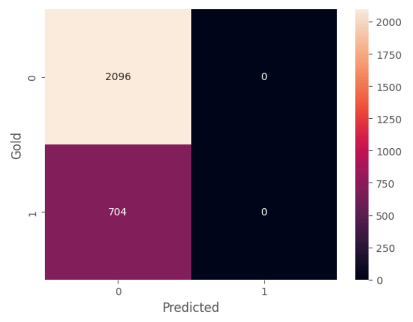
Results and Analysis:

Task A

LSTM Model Test Accuracy 0.8336
LSTM Model Test Precision: 0.8051
LSTM Model Test Recall: 0.4460

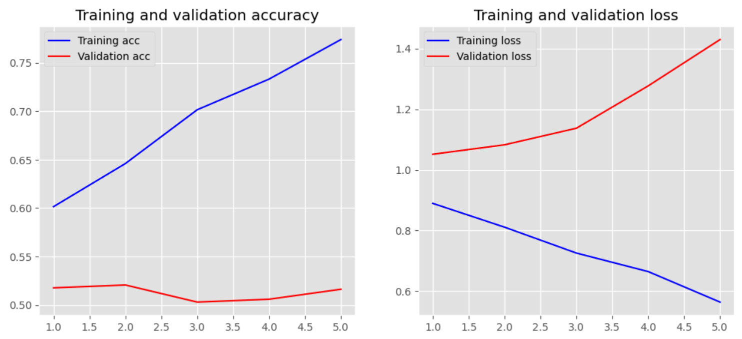






As we have low non sexist data, the model can't detect sexist as much as not_sexist. That's why we have low recall and high precision. The model is also having an overfit problem as we can see from the graph. Training loss is decreasing but testing loss is increasing after a certain point.

Task B
LSTM Model Test Accuracy 0.5176
LSTM Model Test Precision: 0.5361
LSTM Model Test Recall: 0.4044



The problem of overfitting is more severe in task B. The loss doesn't decrease significantly. The lack of data is creating this problem and we also have the problem of imbalanced data.

Overall phase-by-phase improvement:

- In data preprocessing we can use smote to oversample the data. However, oversampling can make the model overfit. We need to handle that using regularization. We can also undersample them but we will lose data in this case.

- We can make the model more complex by adding more layers but we can have an overfitting problem which needs to be handled by regularization.

- The use of transformers can make the classification significantly well

Conclusion:

In conclusion, the sexism detection project has demonstrated the potential of machine learning and natural language processing techniques in identifying and categorizing instances of sexism within textual data.