BRAC University

Date of Submission (9th May, 2024)

Spring 2024

Explainable Detection of Online Sexism (EDOS)

CSE440: Natural Language Processing

Group Members:

Name: Kazi Zunayed Quader Knobo ID: 20241020

Name: Mohammed Sharraf Uddin ID: 20241018

Table of Contents

Dataset Description

Source

The dataset is taken from SemEval 2023.
Link: https://codalab.lisn.upsaclay.fr/competitions/7124

Description

The dataset consists of five columns: rewire_id, text, label_sexist, label_category, label_vector. The dataset shows us what kind of texts are labeled sexist and if sexist what kind of comment it is. The lable_sexist has only two class, which is used for task - A, and label_category has four class, which is used for task- B

Task - A

Dataset Pre-processing

For task - A, except for the text and label_sexist column all the other columns were dropped. The label_sexist initially had two classes: not sexist and sexist. Therefore, unique classes were mapped to numerical values: not sexist was mapped to 0 and sexist was mapped to 1.

The dataset was imbalanced as the occurrence of not sexist was approximately three times more than sexist. This was visualized using a bar chart. To fix this, we created two datasets: one dataset was randomly oversampled and the other dataset was randomly undersampled.

Also, we used the word cloud library to visualize which words were associated with which label.

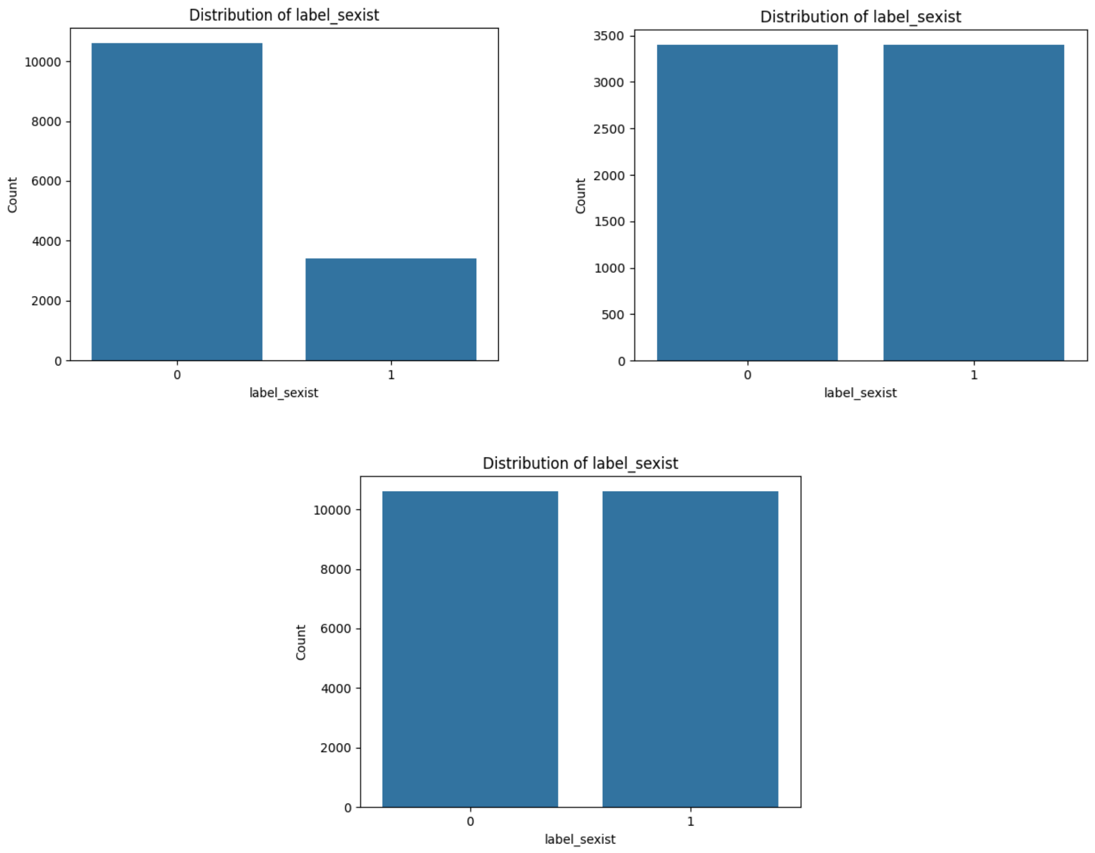


Fig: Imbalanced Dataset, Undersampled Dataset & Oversampled Dataset

Text pre-processing

The text column, in our case the x-variable, was cleaned by removing special characters such as @, !, #, etc. As these special characters do not contain any information, removing them made our dataset less noisy. Moreover, we changed all the alphabets to lowercase and optionally we removed stopwords and performed lemmatization or stemming.

Data splitting, Tokens &Vectors

We split the dataset and kept 80% training and 20% testing, which produced X_train, x_test, y_train and y_test. We tokenized the X_train and x_test to convert words to numbers. These variables were then converted to sequences of tokens and padded so that all the sentences are of the same length. Padding is done so that our neural network model can allocate sentences of various lengths. After padding, we created an embedding matrix using GloVe embedding, which stores pre-trained vectors for different words.

Model creation, training, testing & evaluation

We created three neural networks models: LSTM, Bi-LSTM and GRU. All these models had an embedding layer as the input layer to map words to their corresponding vector and input it to the model. The models have only one hidden layer with 128 nodes and a single node with sigmoid activation function in its output layer. We used the drop out and early stopping regularization methods to prevent the model from over fitting. Also, ADAM was used as the optimizer and binary cross entropy as loss function.

We trained the models on different datasets with different text pre-processing techniques. The best result was found using the dataset that was oversampled, stopwords removed and lemmatized. The performance of the models were pretty similar and while evaluating the models using F1-score, precision and recall it was seen that on average all of the models had a precision of 82%, recall of 92% and F1-score of 87%.

	Precision	Recall	F1-score
LSTM	81.3%	94.1%	87.2%
BI-LSTM	84.2%	91.2%	87.6%
GRU	84.0%	91.4%	87.5%

TASK-B

Dataset-preprocessing

Task-B is a multi-label classification task where a sexist text is further classified into four categories. Hence, features apart from ‘text’, ‘label-sexist’ and ‘label-categories’ were removed from the dataset. Later, the four categories like ‘threats’, ‘dergation’,‘animosity’ and ‘prejudiced discussions’ were mapped to numerical values such as 0, 1, 2 and 3.

For better understanding and visualizing, wordCloud was used to observe the most frequent words that appeared in each of these categories.

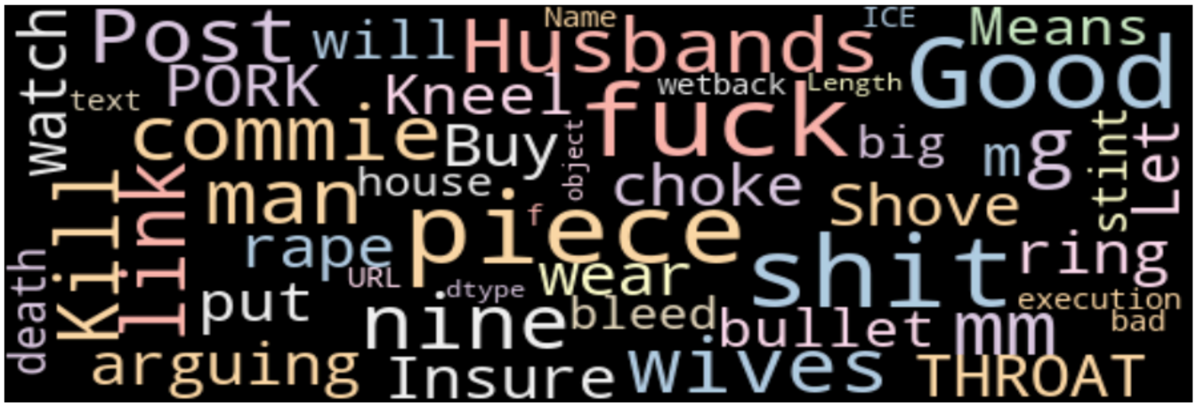
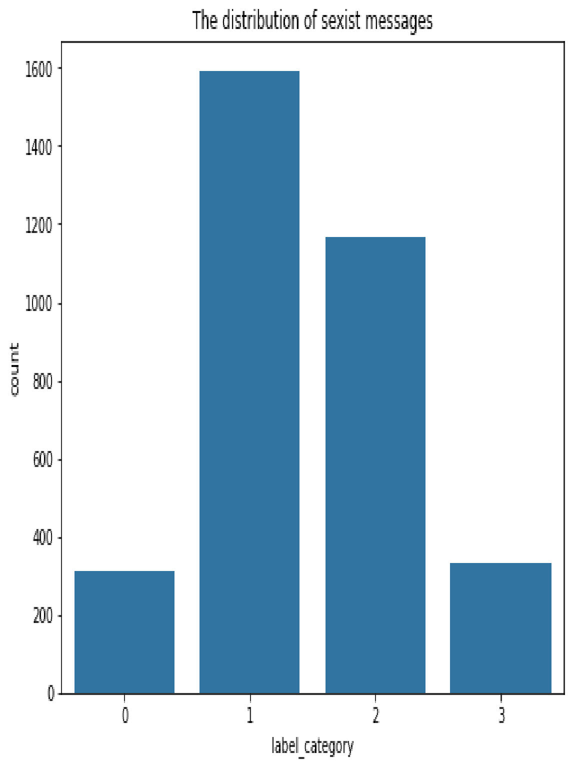
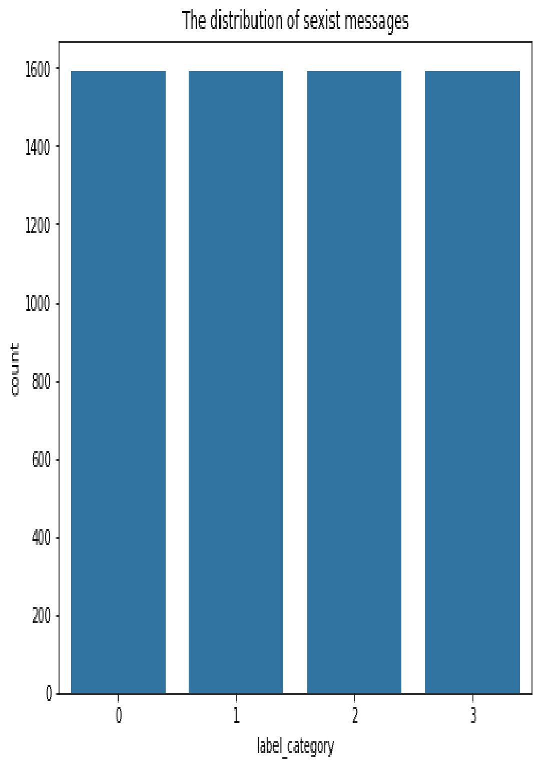


Figure: WordCloud of the category ‘threats’

The Dataset was not balanced, where threats and prejudiced discussions of sexist texts were lower in number. While derogation and animosity texts were greater in amount compared to them. In order to address this issue, we have used Random OverSampling technique to eradicate the imbalance. Below are the bar graphs to show how random oversampling technique has created a balance in the dataset.





Figures: Bar graphs before and after oversampling

<u>Data Splitting & Text-Preprocessing</u>

The sexist text dataset was branched into two parts where 80% of the dataset was for training and 20 % were for testing the model. Thus, x_train, y_train, x_test and y_test were created. Furthermore, for text-processing, Tokenization was introduced in the x_train and x_test dataset. This is where the input text data was converted into numeric representation so that the model can understand. We have used the tokenizer API from tensorflow keras library. The tokenizer will do all the necessary text preprocessing steps like:

- Divide the sentence into words and encode them.
- Specify the maximum number of unique tokens to store in the vocabulary.
- Transform all words into lowercase and all words into integer index.
- Removing all the punctuations and special characters from the texts.
- Longer sentences were truncated to max length which we declared 50 and shorter sentences were padded to length 50 so that all the sentences are of equal length.

Word Embedding

In this step, each word in x_train and x_test is assigned to a high-dimensional vector so that words with similar semantics are closer together in the vector space. For this purpose we have used a very basic technique known as one-hot encoding. In this technique, each word is given a vector representation. The size of the vector is the length of the vocabulary size which is 5000, where all elements in the vector are 0 apart from the index of the word which is set to 1. The output of one hot encoding is passed to the embedding layer where the dimension of the layer is set to 16. Therefore, each word is represented as a 16 dimensional vector.

Model creation, training, testing & evaluation

For multilabel classification of text data into 4 categories we have used two popular gated RNNs: LSTM & GRU. The architecture of both of these recurrent neural networks are similar. The first layer of these recurrent neural networks are the embedding layer for passing each word as a vector input. Both the models have 128 nodes in their respective hidden layers and the activation function used in the output dense layer was softmax. To reduce overfitting, dropout was set to 0.3 meaning 30% of the input units will be randomly turned off during training. Finally, ADAM was our optimizer to reduce loss and the loss function was sparse category cross-entropy.

For performance metrics we have used three measures like precision, recall and F1-score. Below is a table to illustrate a comparison of the performance of LSTM and GRU in classifying sexist texts into 4 categories:

	Precision	Recall	F1-score
LSTM	76%	77%	76%
GRU	80%	81%	80%

From the above table, we can say that GRU performed better in classifying than LSTM. However, overall performance for both of the models are not optimal due to the use of one-hot encoding. Other embedding techniques like word2vec or GloVe might have given better results. This is done deliberately to show how GloVe improves the performance of the model which is implemented in Task-A rather than one-hot encoding in Task-B.