Online Sexism Detection Using Machine Learning and Deep Learning Models

Munia Shaheen
CSE, Brac University
Dhaka, Bangladesh
munia.shaheen@g.bracu.ac.bd

Akib Zabed Ifti
CSE, Brac University
Dhaka, Bangladesh
akib.zabed.ifti@g.bracu.ac.bd

Junaed Hossain
CSE, Brac University
Dhaka, Bangladesh
junaed.hossain@g.bracu.ac.bd

Golam Rabbi
CSE, Brac University
Dhaka, Bangladesh
golam.rabbi@g.bracu.ac.bd

***Abstract*— The prevalence of sexism has increased as social media and internet communication have grown. Online sexism can have negative effects on people, including discomfort, anxiety, and harassment. Therefore, it is now more crucial than ever to create an automatic system that can spot instances of sexism in online conversation. We analyze existing machine learning and deep learning methods for identifying online misogyny on social media sites in this project. To categorize incoming remarks as sexist or not, our approach combines lexical, syntactic, and semantic data. We analyze our model's performance using a test set of comments, and we achieve an F1 score of 89%.**

Keywords—sexist, natural language processing, classification

I. Introduction

Online sexism is becoming a bigger issue. Automated tools are increasingly frequently used to detect and evaluate sexist content on a large scale, however, most of them just classify content into general, high-level categories without providing any further details. The capacity to identify sexist content and to explain why it is sexist enhances the interpretability, trust, and comprehension of the choices made by automated systems, giving users and moderators more control. In order to identify racism and sexism, we utilized a variety of models in this study, including Logistic Regression, AdaBoost, K Neighbors Classifier, Bernoulli Naive Bayes, Multinomial Naive Bayes, SVM, Random Forest, and Deep Neural Network-based classifiers. We will use the F1-score for evaluation and pick the best model for our task.

II. Dataset

We utilized a dataset of 14000 rows and 5 columns of text messages. Review id, text, label sexism, category, and vector were all contained in the columns. Since detecting sexism was a binary classification problem, we simply needed the review id and text fields. Finally, we find out which words are encountered mostly in our dataset which is shown in Figure 1.

III. Methodology

In order to automatically detect online sexism on social media networks, existing machine learning and deep learning models were used. The model was trained using a sizable corpus of annotated data that included both sexist and non-sexist comment instances. To categorize incoming remarks as sexist or not, we combined lexical, syntactic, and semantic data. In order to obtain insight into the linguistic and sociological elements influencing online sexism, the model's performance was assessed on a test set of comments and an analysis of its predictions was carried out.

A. Data Cleaning

- Replace contractions: A contraction in the English language is a word or phrase that has been abbreviated by omitting one or more letters, such as "I'm" in place of "I am". The contractions can be divided ("I'm" to "I "+"'m") or restored to their original form ("I'm" to "I am"). In my experience, the latter is more effective because sub-words like "'m" are more difficult to find a word embedding for.
- Removing the punctuation: Without commas, brackets, or other punctuation, the sentences should be written. A list of punctuation characters is provided by the Python constant string punctuation. This list will be used to remove all punctuation from our text.
- Remove HTML tags: We must remove the HTML tags to make sure there are no invalid data in the dataset.
- Single character removal: When we remove apostrophes from the word "Mark's", the apostrophe is replaced by an empty space. Hence, we are left with the single character "s" that we are removing here.
- Remove multiple spaces: Next, we remove all the single characters and replace them with a space that creates multiple spaces in our text. Finally, we remove the multiple spaces from our text as well.
- Dropping columns: Review id, label category, and vector columns were all dropped as detecting sexism was a binary classification problem and these columns were unnecessary.

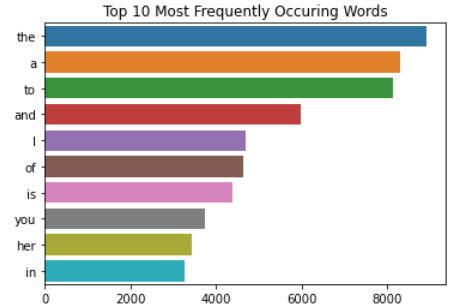

Fig 1: 10 most frequently used words

B. *Tokenize and Padding*

In the majority of NLP tasks, each word in the text must be represented by an integer value (index) before being fed to any model. The text will be changed into a series of integer values in this manner. We will use a vectorizer to convert a collection of text documents to a matrix of token counts for the basic ML models.

The text can be tokenized using a Keras API for the DL models. Each different word's frequency is determined by the Keras tokenizer, which then arranges the words according to that frequency. By converting each text into either a series of integers (each integer representing the index of a token in a dictionary) or a vector with a binary coefficient for each token depending on word count and based on TF-IDF, this enables the vectorization of a corpus of texts.

All your input sequences to the model need to have the same length. In order to achieve that, we can use a function that pads the short sequences with zeros (options are 'pre' or 'post' which pads either before or after each sequence). This function transforms a list (of length num_samples) of sequences (lists of integers) into a 2D Numpy array of shapes (num_samples, num_timesteps). we are using the max-len parameter of the function. While using the max-len parameter, the pad_sequences function shows the array of the same length we have defined in the value of max-len arguments.

C. *Oversampling*

In order to balance the class distribution in unbalanced datasets, we have used oversampling utilizing SMOTE (Synthetic Minority Over-sampling Technique), a technique that creates synthetic samples of the minority class.

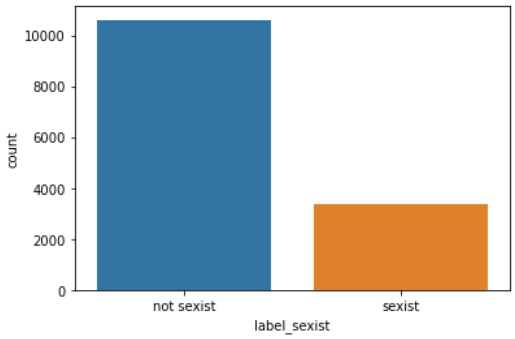

Fig2: Target column before oversampling

By interpolating between already existing minority samples, SMOTE creates new synthetic minority samples. By choosing adjacent pairs of minority samples, fresh synthetic samples are created along the line segment joining them to do the interpolation. Machine learning models have been found to perform better on unbalanced datasets when using the SMOTE method. It should be emphasized, nevertheless, that excessive sampling can result in overfitting and poor generalization, especially if the synthetic samples are too similar to the existing samples.

D. *Word Embedding*

Now that we have tokenized the text, we use GloVe pre-trained vectors. Word embeddings are a way to represent words with similar meanings to have an equal representation. Using word embedding through GloVe, we can perform well with models with even relatively small label training sets.

E. *Models*

We split our data into train and test data and used Logistic Regression, AdaBoost, K Neighbors Classifier, Bernoulli Naive Bayes, Multinomial Naive Bayes, SVM, and Random Forest. Then using the embedding matrix based on GloVe vectors, we can design a model and pre-train the embedding layer with it. The embedded layer is the top layer. The Keras embedding layer is a versatile layer that can be utilized with or without weights that have already been trained. In that instance, the Embedding layer will learn an embedding for each word in the training dataset after being initialized with random weights. In this experiment, the embedding layer's weights will be set to the embedding matrix derived from the GloVe pre-trained vectors. This learning is transferable. The "trainable" parameter in the Embedding layer can be set to True if we want to fine-tune the word embedding or False if we don't want the embedding weights to be updated. In this case, we set it to False. A regularization-focused Dropout layer, an LSTM, GRU, and bidirectional LSTM are placed after the Embedding layer. Our validation split for each deep learning model was 0.2, batch size was 128 and epochs were 50. A dense layer with sigmoid activation transforms the output of the preceding layers into 0 or 1 (sexist or not sexist), and we employed the Adam optimizer in our models.

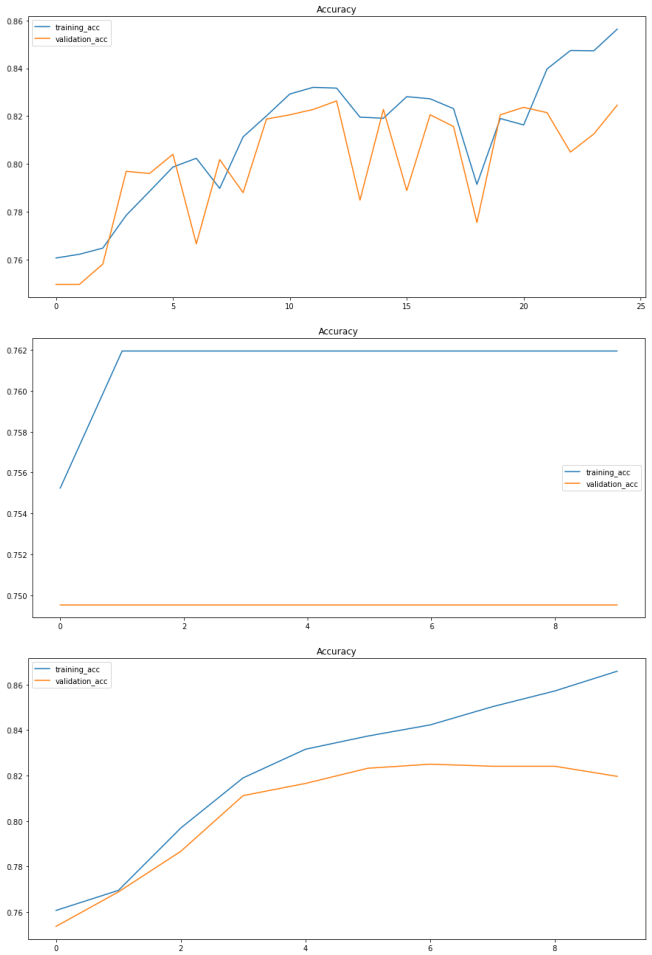

Fig3: Accuracy f LSTM, GRU, and Bi-LSTM model

F. Results

To evaluate the effectiveness of our proposed sexism detection approach, we conducted experiments on a benchmark dataset consisting of 14,000 texts annotated with sexism labels by human experts. First, we evaluated the efficacy of various sexism detection machine learning algorithms, such as Logistic Regression, AdaBoost, K Neighbors Classifier, Bernoulli Naive Bayes, Multinomial Naive Bayes, SVM, and Random Forest. Our tests revealed that Multinomial Naive Bayes outperformed the other models, with an F1 score of 0.83. We further compared the performance of the LSTM, GRU, and Bi-LSTM models with several baseline methods, including logistic regression, decision tree, and support vector machine (SVM). Our experiments showed that the LSTM model outperformed all baseline methods with an F1 score of 0.89.

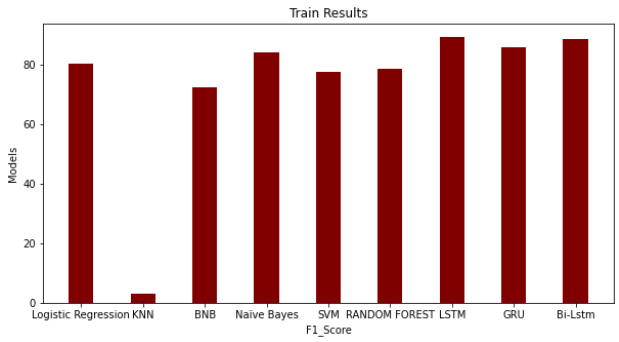

Fig4: F1-score of all the used models

Overall, we can say for ML Multinomial naive bayes works pretty well than others, and for DL LSTM works better than other deep learning models.

Also, our experiments demonstrate the effectiveness of our proposed approach for detecting sexism in social media and highlight the importance of using a combination of lexical, syntactic, semantic, and user features to achieve high performance.

Result	
Models	***F1-score***
Logistic Regression	*0.8036*
AdaBoost	0.7731
AdaBoost with Logistic Regression	0.7620
KNN	0.0299
BernoulliNB	0.7230
MultinomialNB	0.8399
SVM	0.7846
Random Forest	0.7326
LSTM	0.8913
GRU	0.8562

Result	
Models	***F1-score***
Bi_LSTM	0.8850

Fig5: F1-score of all the used models

G. Conclusion

In this research, we showed how machine learning and deep learning models might be used to automatically detect online misogyny on social media platforms. We created a model that used a combination of lexical, syntactic, and semantic variables and was trained on a sizable corpus of annotated data to categorize new comments as sexist or not. Our LSTM model outperformed other machine learning models, achieving an F1 score of 89%. To learn more about the linguistic and social aspects influencing online sexism, we also examined the model's predictions. Our findings emphasize the necessity of creating efficient tools for spotting and stopping dangerous online communication behavior. To investigate how well the model generalizes and expands to other languages, more research is required.

ACKNOWLEDGMENT

We would like to acknowledge the contribution of our course instructors for their guidance and support throughout the semester. We are also grateful to the annotators for their effort in creating the annotated dataset.

REFERENCES

[1] *CodaLab - Competition*. (n.d.). https://codalab.lisn.upsaclay.fr/competitions/7124

[2] Sepidafs. (2020). Text Classification with GloVe, LSTM/GRU(99% acc). Kaggle. https://www.kaggle.com/code/sepidafs/text-classification-with-glove-lstm-gru-99-acc/notebook

[3] The AI & DS Channel. (2022, March 31). *Twitter Hate speech detection using machine learning | Machine Learning Project 9* [Video]. YouTube. https://www.youtube.com/watch?v=TLbb1UbU0aQ

[4] Taha Junaid. (2020, September 22). *NLP-Hate Speech Detection using LSTM in Tensorflow* [Video]. YouTube. https://www.youtube.com/watch?v=2qnXmB65Ino

[5] Rithesh Sreenivasan. (2020, September 11). *Text Classification using LSTM on Amazon Review Dataset with TensorFlow 2.1 #nlp #tutorial* [Video]. YouTube. https://www.youtube.com/watch?v=DUZn8hMLnbI

[6] Nissa, N. K. (2022, August 23). Text Messages Classification using LSTM, Bi-LSTM, and GRU. *Medium*. https://medium.com/mlearning-ai/the-classification-of-text-messages-using-lstm-bi-lstm-and-gru-f79b207f90ad#:~:text=A%20Gated%20Recurrent%20Unit%2C%20or,train%20than%20their%20LSTM%20counterparts.