Module: Applied Artificial Intelligence
Module code: 771767
Student code: 202125489
Assignment Topic: AI-based offensive text detection system

Table of Contents

Abstract

Offensive and hateful posts have become a critical problem in the social media domain. Political uprisings, insurrections, and bullying have been carried out using social media platforms. Hence, a system that automatically detects these inciteful posts and flags them before escalation is paramount. Artificial intelligence is becoming state of the art (SOTA), the latest being chat-GPT3 a conversational bot built on generative pre-trained transformer deep learning architecture. Algorithms like this can be harnessed in developing the SOTA classification system. This project focuses on implementing a deep learning system for text classification (offensive language detection). In this project, context-independent word embedding algorithms (word2vec, GloVe, and FastText) were used with BiLSTM a variant of RNN models. The BERT model (a context-dependent algorithm) was also implemented, and the performance of the systems was evaluated and compared. The GloVe + BiLSTM and BERT algorithms scored the best (95%) out of all after being trained on tagged tweets from Twitter. The project aims to foster discussion and works on natural language processing within the research community.

***Keywords**: offensive language, text classification, deep learning*

Introduction

Freedom of expression is a fundamental human right as declared by the United Nation General Assembly held in the city of Paris in 1948 (UNGA, 1948) Freedom of expression does not give one the right to discriminate against others hence laws like the New York law against discrimination which prohibits discrimination, based on color, race, religion, or nationality (Berger, 1952). The emergence of social media has given people much freedom to express their opinion but unfortunately, some bad actors have abused this privilege and leveraged it to spread hate and abuse. The rise and scale of social media make it possible to reach a large audience in record time hence it can be dangerous when used negatively like spreading hate. For example, the Capitol Hill riot during the US 2021 elections (BBC, 2021), was sparked by the Twitter post of Donald Trump.

Automatically detecting offensive text has gained increased attention from researchers and with the advancement in artificial intelligence, especially natural language processing, the development of algorithms is taking shape. Detection of offensive or toxic content is a text classification task known as "sentiment analysis" in the field of NLP.
This project utilizes word embedding techniques, as opposed to the classical approach which converts text to plain numbers before fitting on classifiers. Rather, the corpus of text is mapped to high-dimension vectors based on the context of words and their semantic meaning aimed at building models that can perform well in classifying offensive textual content.
The expected outcome for the project is the development of a highly accurate and efficient model for detecting offensive textual content using word embedding techniques. The model should be capable of detecting offensive textual content and perform well when deployed to live applications. This project will contribute to the research on natural language processing and provide insights into the use of context-dependent and context-independent word embedding techniques in text classification.

The rest of this report will cover the research background, objective, methodology, experiments, and results.

Background

Offensive text detection is a text classification task and is hence considered supervised learning in the field of artificial intelligence. Research on text classification generally utilizes either machine learning models like Support Vector Machine (SVM) (Waseem & Hovy, 2016) or deep neural network (DNN) architectures (Zhang, et al., 2018). In the classical approach, features engineering is manually done on the dataset before fitting it on the models (Burnap & Williams, 2015) while in the DNN approach, the neural network learns the features automatically during training (Gamback & Sikdar, 2017).
Dealing with textual data requires a different approach from numerical data as computer models cannot process text directly but numbers hence the need to parse non-numerical data (text) into numerical data, the process which is known as vectorization. One of the techniques in vectorization is Term Frequency Inverse Document Frequency (TFIDF) which focuses both on the frequency of

the words or n-grams and the importance of the token by factoring the weight too. Tokens that appear in many documents are considered less important than those that appear in a few documents. (Dinakar, et al., 2011) in their research on detecting cyberbullying in text, they used the TFIDF feature representation technique in vectorizing the textual data and Support Vector Machine as the classifier model. They achieved an accuracy of 79% in their analysis. Much more recent work was done by (Abro, et al., 2020) using the same vectorization technique and the JRip model which is a rule-based model that uses sequential covering had an accuracy of 73%. The analysis was on detecting hate speech in textual data.

Another vectorization technique is bag-of-words (BOW) in which the textual data is represented by a vector of word frequencies (occurrence of each word or n-gram in the document). This technique was used in combination with TFIDF by (Koushik, et al., 2019) to automatically detect hate speech on Twitter. They achieved an accuracy of 94% using the Logistic regression model as the classifier.

The vectorization techniques used in these works have limitations in performance due to their inability to understand the semantics and context of the words being vectorized. Modern techniques have now been developed to handle the limitations of these count-based techniques. These are known as word embeddings. It represents text as high-dimensional vectors which capture the semantic, syntactic, and contextual meaning of each token in the document. Examples of algorithms that utilize the word embeddings technique include word2vec, GloVe, and FastText. In their research on hate speech detection on Twitter, (Robinson, et al., 2018) compared the performance of models with carefully engineered features using ML algorithms (SVM) and automatic feature learning using DNN models (CNN and GRU). In the research, they were able to archive high performance using DNN automatic feature learning approach than with the ML algorithm.

DNN's ability in processing hierarchical representations of data makes it ideal for learning high-dimension vector output from word embedding models. DNN can be trained on these vectors that have captured the text's meaning and context. DNN architectures include CNN and RNN.

This research explores the performance of word embedding techniques (word2vec, GloVe, FastText) with RNN classifiers and transformer architecture (BERT) and compares them with traditional techniques and machine learning models on the classification of text.

Objective

The objective in this project is to build a text classifier based on deep learning (RNN and pre-trained transformers) and word embedding techniques. A classifier system that can tag a text as offensive or not. Several word embedding techniques will be explored, and the performance of the classifier models will be evaluated and compared.

Methodology

This section describes the suggested system that has been used to classify text into "offensive" or “not offensive." The whole research approach is shown below (Figure 1). The research methodology is divided into six phases which are data collecting, data preprocessing, visualization, vectorization, model architecture, and model evaluation. The following sections go into detail on each phase.

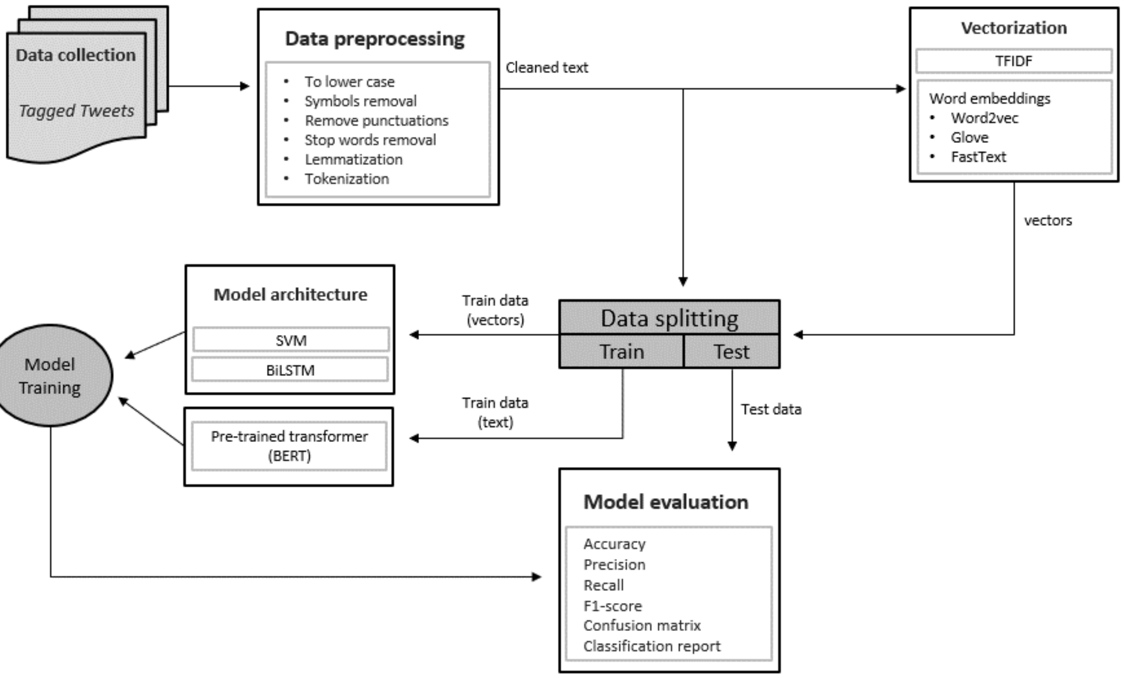


Figure 1. System overview

Step 1: Data collection
Two datasets were used in this research namely “tweet sentiments” (Yasser, 2022) and “Twitter hate and offensive language” (Samoshyn, 2020). These are publicly available datasets obtained from Kaggle and have been labeled accordingly. Details of the datasets will be discussed in the visualization phase of this research.

Step 2: Data Preprocessing
The data collected is then preprocessed in a clean and consistent format. This involves tasks such as lower-case conversion, punctuation removal, special characters removal, stop-word removal, and lemmatization. In removing the punctuations and special characters, regular express (RegEx) was employed to remove items matching the patterns specified. This also removed usernames, mentions, hashtags, and white spaces. For stop word removal, NLTK’s stopwords library was used to remove stop words as they do not carry meaning in the context of natural language processing. Lemmatization was chosen over stemming given that it retains the base meaning of each word. WordNetLemmatizer, a library from NLTK was employed in lemmatizing the words in the text. The output from preprocessing is a less noisy text with informative features that a classifier can learn during training.

Step 3: Data visualization

The preprocessed data is then visualized. The distribution of the classes is shown to check for the balance in the classes. An imbalance in the classes will lead to the model biasing towards the majority class. Figure 2 shows the class distribution in the "Twitter hate and offensive language" dataset. This shows that the class is grossly imbalanced and may affect the model performance. As a result, a second dataset ("tweet sentiments") was collected and its class distribution is shown in Figure 3.

Figure2: dataset1 class distribution

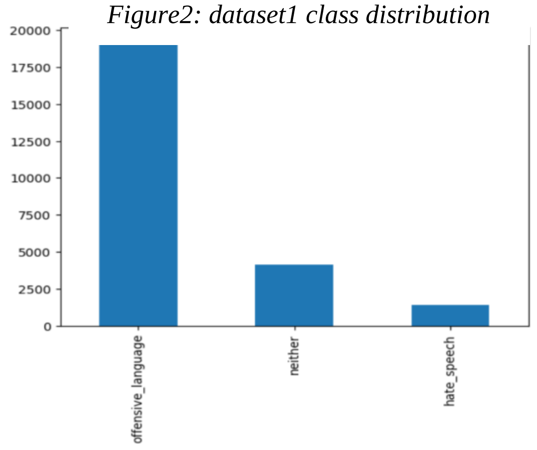
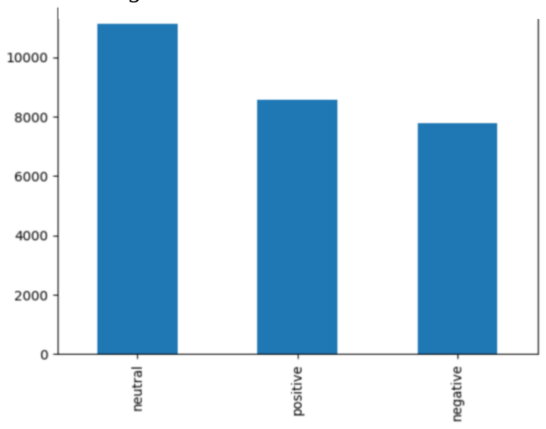
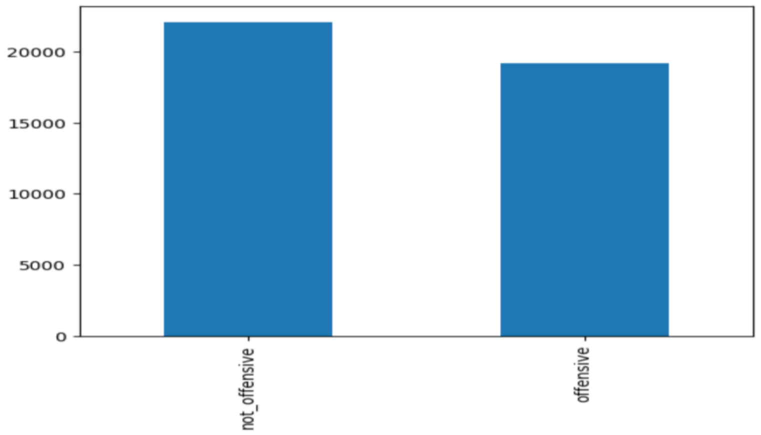


Figure 3: dataset2 class distribution



The "*offensive language*" and "*hate speech*" *classes* are combined with over 16000 random samples from *neutral* and *positive* classes selected from dataset2, merged with dataset1 to obtain a new dataset as shown in figure4.

Figure 4: dataset3 class distribution



As shown in figure 4, the new dataset obtained has a seemly balanced class hence the classifier model biasing towards any class has been curtailed. The "*offensive*" class is the combination of "*offensive language*" and "*hate speech*" while the *"not offensive*" is the combination of *"positive" and "neutral"* classes.

The top occurring words for each class are visualized using word cloud visualization as shown in figure 5 and 6.

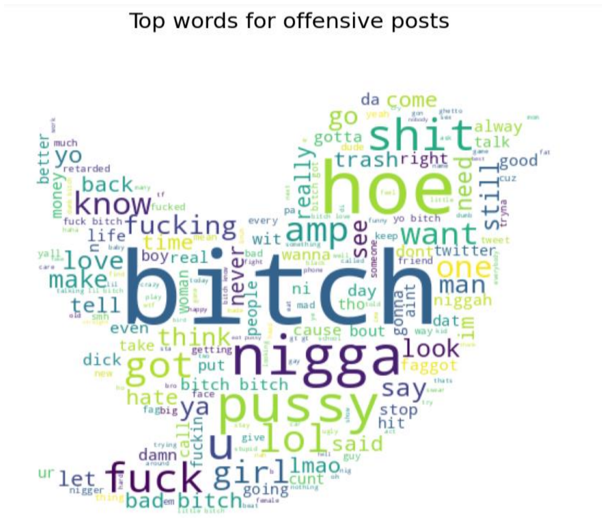


Figure 5: top offensive words

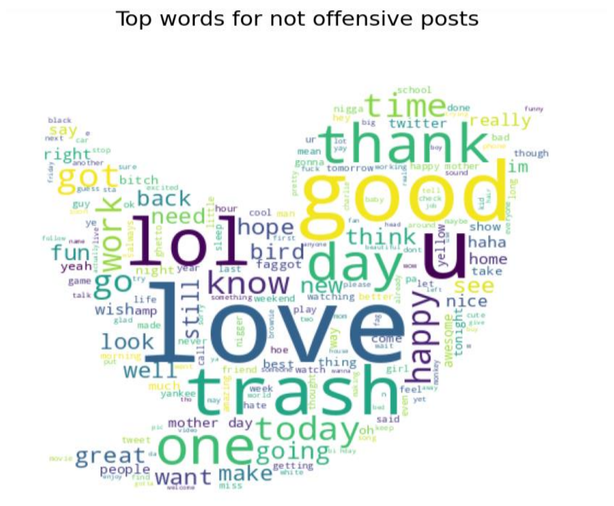


Figure 6: top not offensive words

Step 4: vectorization

In this phase, the cleaned text was converted into numerical formats using word embedding algorithms. TFIDF technique was used to vectorize the data for the baseline model (SVM). The algorithms that utilize the word embeddings technique used in this research are word2vec, GloVe, and FastText. Word2Vec is a vectorization technique proposed by (Mikolov, et al., 2013) for the vectorization of large datasets. The model is less computationally expensive and performs better than previous techniques. GloVe, designed by researchers at Stanford University, combines global matrix factorization and local context window, making it outperform related algorithms on similarity tasks (Pennington, et al., 2014). FastText seeks to handle the limitation of other models on an unlabeled corpus (Bojanowski, et al., 2017).

These algorithms produce high-dimensional vectors for the neural network classifiers.

Step 5: Model architecture

The model architecture for the classifiers is built. Three models were used in this research namely, SVM (the baseline model), BiLSTM, and pre-trained transformer (BERT).

BiLSTM consists of two LSTM layers. One of the layers reads the input in the forward direction and the other layer reads in the reverse direction. This enables the network to capture both the past and the future context of the input data. These two layers are then concatenated at the output and a dense layer is attached to generate a final output.

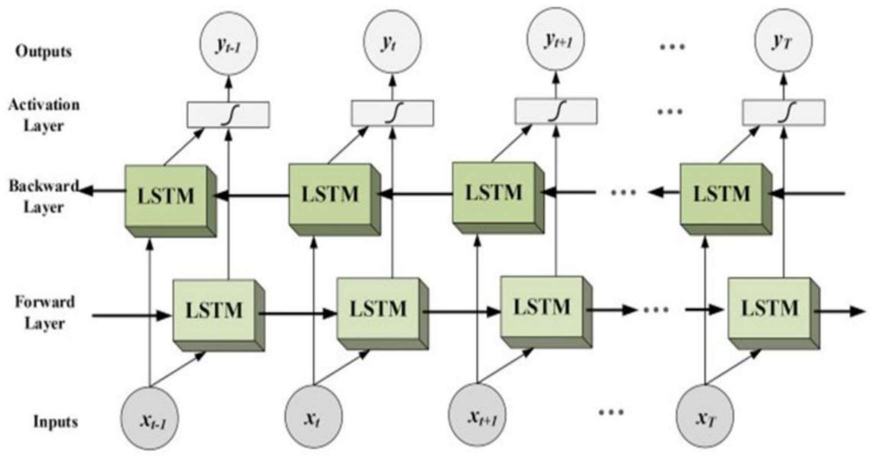


Figure 7: Bi-LSTM image source

Transformer is a model architecture that utilizes attention mechanism (Vaswani, et al., 2017). its input sequence is differentially weighted unlike in convolutional or recurrent models. In this research, BERT is used. It utilizes attention mechanism (transformer) to learn the contextual meaning of words relative to the sentence they are used in when generating word embeddings, rather than just looking at the preceding or following words as traditional NLP models do (Devlin & Chang, 2018). This allows BERT to better capture the meaning and nuances of natural language. The BERT model is a pre-trained model which has been trained on an enormous corpus of text, including the entire Wikipedia corpus and the BookCorpus dataset, using a masked language modeling objective.

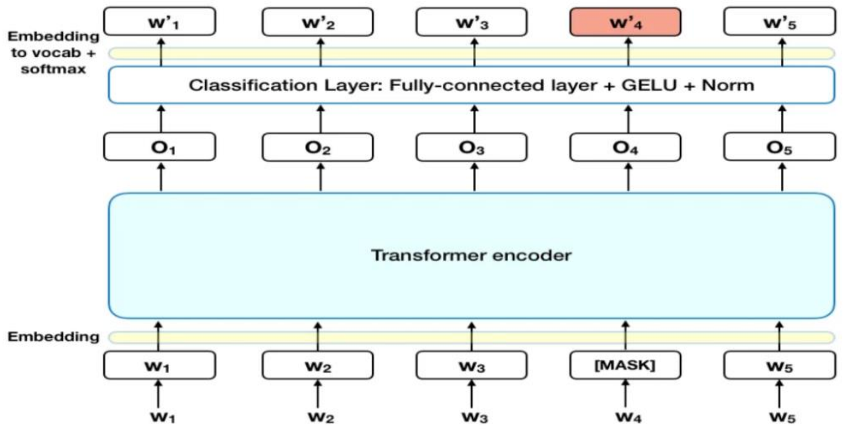

w'1
w'2
w'3
w'4
w'5
Embedding to vocab + softmax
Classification Layer: Fully-connected layer + GELU + Norm
O1
O2
O3
O4
O5
Transformer encoder
Embedding
w1
w2
w3
[MASK]
w5
w1
w2
w3
w4
w5

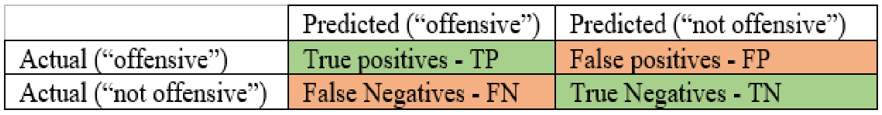
Figure 8: BERT image source

The pre-trained BERT model was accessed from TensorFlow hub. The model has four transformer blocks (L=4), hidden size (H=512), and attention heads (A=8). For this research task, the model is fine-tuned by connecting the output to a dense layer for classification.

The model is then fitted on the training dataset and optimized using the testing dataset.

Step 6: Model Evaluation

After the model has been trained, it is evaluated using the test set. It predicts the class ("offensive" or "not offensive") of the test set. The performance is evaluated by obtaining the confusion metrics of the result as shown in Table 1 below.



	Predicted ("offensive")	Predicted ("not offensive")
Actual ("offensive")	True positives - TP	False positives - FP
Actual ("not offensive")	False Negatives - FN	True Negatives - TN

Table 1: confusion matrix

The models' performances were evaluated on accuracy, precision, recall, and F1 measure.

Accuracy: This is the proportion of accurate predictions to all predictions.

$$\text{accuracy} = \frac{TP+TN}{TP+FP+FN+TN}$$

Precision: It is the proportion of expected positives that really turn out to be positive.

$$\text{precision} = \frac{TP}{TP+FP}$$

Recall: This refers to the ratio of correctly predicted positives out of the actual positives.

$$\text{recall} = \frac{TP}{TP+FN}$$

f1 measure: It is the harmonic mean of precision and recall.

$$\text{F1 measure} = \frac{2*recall*precision}{recaall+precision}$$

Some hyperparameters like the learning rate, the number of nodes, and the dropout rate are tuned to improve the performance of the models if necessary.

Experiment

As mentioned in the methodology, four vectorization techniques were used namely TFIDF, word2vec, GloVe, FastText. There is a total of four sets of vectors from the outputs of these techniques. The dataset used is the final dataset obtained after the merging of dataset1 and the subset of dataset1 resulting in a dataset balanced by class as described in the methodology. The train-test split ratio used is 80:20.

Three model architectures were used for the classification, namely SVM, BiLSTM, and fine-tuned BERT (pre-trained transformer). The SVM model served as the baseline model in addition to models from related works. The vectors generated from word2vec, GloVe, and FastText algorithms were passed to the BiLSTM model for training and evaluation.

For the BERT architecture, the BERT model was fine-tuned by stacking it into a network of dense layers. The input to the model is the cleaned text.

Figures 10 and 11 show the model architecture of the classifiers.

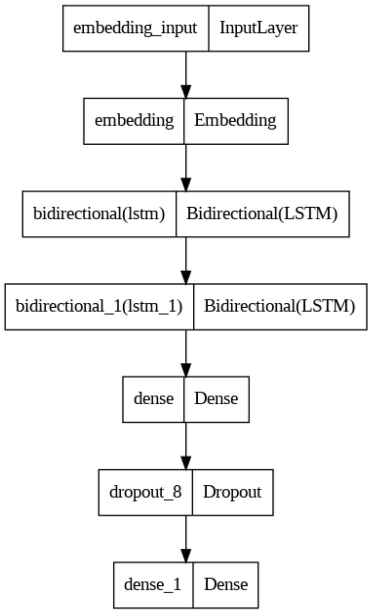

embedding_input | InputLayer
embedding | Embedding
bidirectional(lstm) | Bidirectional(LSTM)
bidirectional_1(lstm_1) | Bidirectional(LSTM)
dense | Dense
dropout_8 | Dropout
dense_1 | Dense

Figure 9: BiLSTM classifier architecture

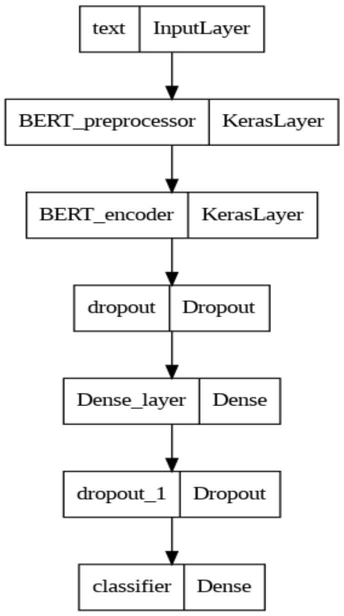


Figure 10: BERT classifier architecture

Dropout layers of 20% were used to enable the models to generalize well. The BiLSTM model was compiled with a learning rate of 1e10-3 while the fine-tuned BERT was 3e10-5 (the model performed poorly when higher values were used). The baseline model for this research is SVM (models from related works) and the dataset was vectorized using TFIDF. The models' accuracy, precision, recall, and f1-score metrics were compared to evaluate their performances.

Results

Table 2 shows the performance metrics of the analyses. Each metric's top outcome is boldened, and the baseline result is highlighted. In all, the BiLSTM model using word2vec and FastText scored the lowest (89%) in all the metrics analyzed while BERT and GloVe scored the highest (95%) followed closely by TFIDF which scored 94%. Table 3 shows other baselines. These are from the literature reviewed.

model		SVM	BiLSTM			BERT
embedding technique		TFIDF	word2vec	GloVe	FastText	
metric	Accuracy	0.94	0.89	**0.95**	0.89	**0.95**
	Precision	0.94	0.89	**0.95**	0.89	**0.95**
	Recall	0.94	0.89	**0.95**	0.89	**0.95**
	F1-score	0.94	0.89	**0.95**	0.89	**0.95**

Table 2: performance metrics of the models

Research	Embedding technique	Model	Accuracy achieved
Hate Speech Detection (Abro, et al., 2020)	TFIDF	JRip classifier	73%
Automatic Detection of Hate Speech on Twitter (Koushik, et al., 2019)	BOW + TFIDF	Logistic regression	94%

Table 3: Other baselines

The confusion matrix of the best-performing analysis is shown in Figure 11-12. Figure 11 shows that BiLSTM using GloVe, out of 3838 tweets labeled offensive, 3618 were classified correctly by the model while out of 4410 labeled as not offensive, 4208 were classified correctly. Figure 12 shows that the BERT model classified 3615 tweets correctly as offensive and 4185 as not offensive correctly.

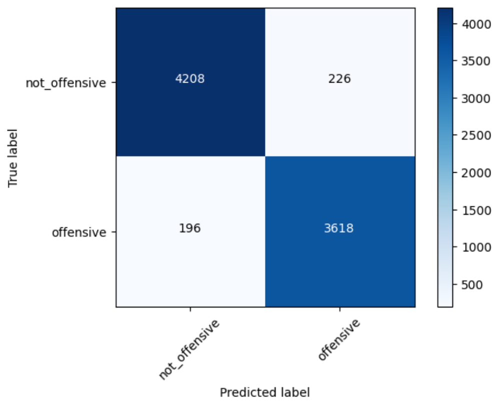


Figure 11: embedding: GloVe, classifier: BiLSTM

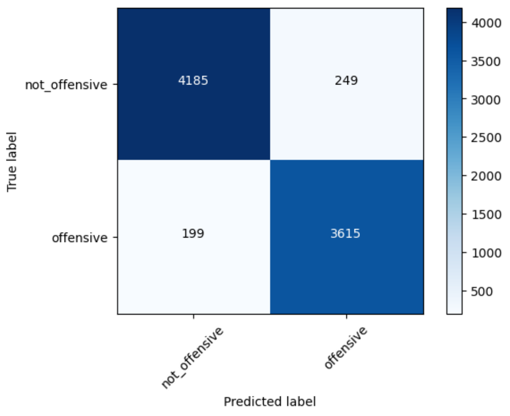


Figure 12: BERT

The experiment result showed that the BERT and GloVe models achieved the best performance. While the margin of performance is not wide, BERT is a state-of-the-art model having pre-trained on an enormous body of text and its attention mechanism. This gives it an edge on way more complex NLP tasks. The GloVe word embedding technique performed well possibly because it captures the global occurrence statistics of words within the corpus of text. The GloVe model used was glove-twitter-100 which was pre-trained on the Twitter dataset (recall the dataset used for this research is the Twitter dataset)

Conclusion

In conclusion, the goal of this research was to assess how well various word embedding methods and model architectures classified offensive language in tweets. The results showed that BERT and GloVe + BiLSTM achieved the best outcome.

The performance of BERT, which is a model pre-trained on a big body of text, highlights the importance of pre-training in NLP tasks. GloVe, on the other hand, performed well possibly because it captures global occurrence statistics of words within the corpus of text. The results provide insights into the effectiveness of different techniques and models in classifying offensive language in tweets. Most importantly, it showed how pretraining models and then fine-tuning for specific tasks can greatly improve the performance of AI systems in real-life applications.

Future work can explore the effectiveness of other pre-trained models and different word embedding techniques, especially generative pre-trained transformers GPT. Additionally, the study can be extended to other languages and different types of text, such as comments and reviews, to evaluate the generalizability of the findings.

References

Abro, S. et al., 2020. Automatic Hate Speech Detection using Machine Learning: A Comparative Study. *(IJACSA) International Journal of Advanced Computer Science and Applications,* 11(8).

Anderson, D., 2017. *Independent report: Attacks in London and Manchester between March and June 2017.* [Online]
Available at: https://www.gov.uk/government/publications/attacks-in-london-and-manchester-between-march-and-june-2017
[Accessed 6 March 2023].

Baroni, M., Dinu, G. & Kruszewski, G., 2014. *Don't count, predict! A systematic comparison of context-counting vs. context-predicting semantic vectors.* s.l., Association for Computational Linguistics (ACL).

BBC, 2021. *BBC News.* [Online]
Available at: https://www.bbc.co.uk/news/world-us-canada-56004916
[Accessed 6 March 2023].

Berger, M., 1952. IV: The New York State Law Against Discrimination: Operation and Administration. In: *Equality by Statute: Legal Controls Over Group Discrimination.* New York: Columbia University Press, pp. 109-169.

Bojanowski, P., Grave, E., Joulin, A. & Mikolov, T., 2017. *Enriching Word Vectors with Subword Information.* New York, Cornell University.

Brown, H., Guskin, E. & Mitchell, A., 2012. *The Role of Social Media in the Arab Uprisings,* Washington: Pew Research Centre.

Burnap, P. & Williams, M. L., 2015. Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making. *Policy and Internet,* 7(2), pp. 223-242.

Devlin, J. & Chang, M.-W., 2018. *Open Sourcing BERT: State-of-the-Art Pre-training for Natural Language Processing.* [Online]
Available at: https://ai.googleblog.com/2018/11/open-sourcing-bert-state-of-art-pre.html?m=1
[Accessed 3 4 2023].

Dinakar, K., Reichart, R. & H, L., 2011. *Modeling the detection of textual cyberbully.* s.l., Association for the Advancement of Artificial Intelligence (AAAI).

EU, 2019. *European Commission.* [Online]
Available at: https://commission.europa.eu/strategy-and-policy/policies/justice-and-fundamental-rights/combatting-discrimination/racism-and-xenophobia/eu-code-conduct-countering-illegal-hate-speech-online_en
[Accessed 6 March 2023].

Fortuna, P. & Nunes, S., 2018. A Survey on Automatic Detection of Hate Speech in Text. *ACM Computing Surveys,* Volume 51, pp. 1-30.

Gamback, B. & Sikdar, U., 2017. *Using convolutional neural networks to classify hate speech.* s.l., s.n., pp. 85-90.

Greevy, E. & Smeaton, A., 2004. *Classifying racist texts using a support vector machine.* s.l., Association for Computing Machinery (ACM).

Koushik, G., K, R. & Muthusamy, S., 2019. Automated Hate Speech Detection on Twitter. *2019 5th International Conference On Computing, Communication, Control And Automation (ICCUBEA),* september.

Kwok, I. & Wang, Y., 2013. *Locate the hate: Detecting tweets against blacks.* s.l., Association for the Advancement of Artificial Intelligence (AAAI).

Mikolov, T., Chen, K., Corrado, G. & Dean, J., 2013. *Efficient Estimation of Word Representations in Vector Space,* New York: Cornell University.

Nockleby, J. T., 2000. *Hate Speech.* New York: Macmillan.

Pennington, J., Socher, R. & Manning, C. D., 2014. *GloVe: Global Vectors for Word Representation.* s.l., Research Gate.

Robinson, D., Zhang, Z. & Tepper, J., 2018. *Hate Speech Detection on Twitter: Feature Engineering vs Feature Selection.* Crete, Greece, Springer, pp. 46-49.

Samoshyn, A., 2020. *Hate Speech and Offensive Language Dataset.* [Online]
Available at: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset
[Accessed 04 March 2023].

Tulkens, S. et al., 2016. *A dictionary-based approach to racism detection in dutch social media.* s.l., Language Resources and Evaluation Conference (LREC), pp. 11-17.

UNGA, 1948. *The Universal Declaration of Human Rights (UDHR).* New York, General Assembly United Nations.

Vaswani, A. et al., 2017. *Attention is All you Need.* [Online]
Available at:
https://proceedings.neurips.cc/paper_files/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html
[Accessed 21 April 2023].

Waseem, Z. & Hovy, D., 2016. *Hateful symbols or hateful people? predictive features for hate speech detection on Twitter.* s.l., Association for Computational Linguistics, pp. 88-93.

Yasser, M., 2022. *Twitter Tweets Sentiment Dataset.* [Online]
Available at: https://www.kaggle.com/datasets/yasserh/twitter-tweets-sentiment-dataset
[Accessed 04 March 2023].

Zhang, Z., Robinson, D. & Tepper, J., 2018. *Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network.* s.l., European Semantic Web Conference (ESWC).