**HATE SPEECH-OFFENSIVE LANGUAGE DETECTION MODEL**

**PROBLEM STATEMENT**

The rise of social media platforms like Instagram, X(formerly Twitter), Reddit, YouTube etc. revolutionised communication, allowing millions and billions of users to express their opinions and feelings freely. This open environment has also led to significant increase in unchecked spread of Hate speech, Offensive language and Toxic behaviour targeting different cultures, beliefs, countries, classes of people which contributes to unhealthy digital environment especially for young minds.

Manual moderation of contents is no longer feasible due to its sheer volume and speed at which these comments come. Traditional keyword-based filters often fail to detect the nuanced and context dependent nature of speech, particularly when users intentionally hide harmful comments through abbreviations/symbols, slang, or sarcasm.

**RELATED WORK**

Hate speech detection has evolved through three key phases:

**Early Approaches (2017-2019):**  
Davidson et al.'s original work (2017) used TF-IDF with logistic regression (F1=0.51 for hate speech). Gambäck & Sikdar (2017) showed CNNs could improve this to F1=0.55 but struggled with subtle context. The shift to LSTMs (Zhang et al., 2018) brought modest gains (F1=0.53) but retained sequential processing limitations.

**Transformer Era (2020-2022):**  
Al-Hassan et al. (2019) demonstrated BERT's superiority (F1=0.68), while Malik et al. (2022) combined BERT with SVM kernels (macro F1 = 0.72; Class 0 F1 ≈ 0.63). These established that contextual embeddings were crucial but faced precision-recall trade-offs.

**Recent Advances (2023-2024):**  
Mnassri et al. (2023) achieved F1=0.75 using RoBERTa with attention pooling, and while Mozafari et al. (2020) achieved strong macro F1 using DeBERTa-GNN, class-specific scores were not reported, making direct Class 0 comparison infeasible. While powerful, these require either complex architectures (Mnassri) or user network data (Mozafari).

My work bridges this gap by achieving a competitive F1 score of 0.79 using a simpler, deployable BERT-XGBoost pipeline that relies solely on raw text inputs — without complex metadata or user graph features.

**MOTIVATION**

This project addresses the need for a robust, automated speech detection system that not only classifies text into Hate/Offensive/Normal but also adapts to the informal and diverse nature of online discussions.

My naïve approach to this solution integrates the contextual understanding power of **BERT (Bidirectional Encoder Representations from Transformers**) with decision making capabilities of **XGBoost (Xtreme Gradient Boosting)**; aiming to achieve high accuracy especially for Hate speech which is often the hardest to distinguish due to its subtlety and lower representations in datasets.

**TOOLS, LANGUAGE AND LIBRARIES USED**

The following tools and libraries were used in building and evaluating this project:

**Programming Language**: Python 3.10 – used for scripting, data preprocessing, modelling, and evaluation.

**Development Environment**: Visual Studio Code (VS Code) – chosen for its flexibility, extension support, and debugging tools.

**Libraries & Frameworks**:

* transformers (by Hugging Face) – for accessing the pre-trained BERT model (Bert-base-uncased).
* torch (PyTorch-CUDA) – backend for tensor operations and model execution.
* scikit-learn – for model evaluation (classification reports, precision, recall, etc.).
* xgboost – used as a classifier on top of the BERT embeddings due to its robustness and performance.
* nltk and re – for text preprocessing tasks such as stopword removal, lemmatization (reduces words to their base or root form (the lemma) to facilitate analysis), and regular expression-based filtering.
* matplotlib and seaborn – for generating plots and visualizations like confusion matrix and SHAP summary.
* SHAP(optional) – for explainability and understanding model decision behaviour using SHAP values.

**Hardware**: NVIDIA CUDA-enabled GPU RTX3050 – leveraged for accelerated training and inference time.

**DATASET DESCRIPTION**

For this project, I used a well-known *Davidson et al*. Dataset containing thousands of tweets manually labelled into one of three categories:

1. **Class 0** –HATE SPEECH: Tweets that directly express hate or encourage violence against a group of people.
2. **Class 1** –OFFENSIVE LANGUAGE: Tweets that are offensive or abusive, but not necessarily hateful.
3. **Class 2** – NORMAL SPEECH: Tweets that are neutral or free from harmful content.

Link: <https://www.kaggle.com/datasets/eldrich/hate-speech-offensive-tweets-by-davidson-et-al/data>

**Datasets description**. The columns show the total number of tweets, the different categories and the percentage of tweets belong to dataset.



Each tweet is labelled based on its intent and severity, providing an overview of Toxic discourse online.

Preprocessing of Dataset included:

* ULRs, Usernames and punctuations were removed.
* Lemmatisation was applied to reduce words to their root form while preserving their contextual tokens such as **\*** , **@** , **#** which often carry hidden meaning in social media text.
* Attention paid to informal nature of online communication, retaining slangs and quirks typical of platforms like X.

**METHODOLOGY**

To Tackle this problem, considering informal, noisy social media text, a two-step pipeline was used combining the strength of Modern NLP and traditional machine learning.

1. TEXT EMBEDDING WITH BERT

**Bert-base-uncased model** from HuggingFace’s Transformers library to convert tweets into dense vector representations. BERT’S deep bidirectional nature allows it to understand context better than traditional embedding techniques like TF-IDF or Word2Vec. Each tweet is tokenised and transformed into the fixed size 768-dimensional embedding, capturing nuanced contexts and intent.

I chose BERT over LSTM and RNN because BERT can understand the full context of a word through its ability of self-attention capturing long range dependencies, essential for this model while LSTM and RNNs struggle with long range dependencies and need more data and training to reach comparable accuracy.

1. CLASSIFICATION WITH XGBoost

Instead of using typical neural classifier on top of BERT, I opted for XGBoost,

A powerful, tree-based ensemble learning method known for its performance and robustness with tabular data.

This approach not only improved generalisation but also helped in dealing with class imbalance, particularly for CLASS-0: HATE SPEECH, which tends to be underrepresented in the dataset.

My hybrid approach was specifically designed to overcome three limitations in recent work:

1. **Dependency on user metadata** (Mozafari's GNN requirement) by using text-only inputs
2. **Black-box decisions** (Mnassri's lack of explainability) through SHAP analysis
3. **Class imbalance sensitivity** (Malik's SVM recall drops) via XGBoost's for Class 0

While direct comparison requires care (some papers omit Class 0 recall), my model shows good score for Class 0 metrics specifically by estimating other papers scores.

**BENEFITS OF THIS HYBRID APPROACH**:

Early version of development of this model using only BERT’s inbuilt classification head struggled with giving high % on precision and recall for Class 0(Hate).

By separating embeddings and classification, I could:

* Gain more control over classification stage where the model was struggling
* Access to hyperparameter tuning specific to XGBoost
* Gain insight into which specific BERT-derived features contributed most to the classification using SHAP as interpretability tool on the tabular embeddings (uncover which embeddings played the biggest role in identifying hate/offensive speech).

**TERMINOLOGIES**

**Hate Speech**: Language that attacks or demeans a group based on attributes like race, religion, ethnic origin, sexual orientation, disability, gender, culture – Labelled as Class 0.

**Offensive Language**: Includes profanities, slurs, or aggressive tone that may not necessarily target a group – Labelled as Class 1

**Normal Speech**: Benign, non-offensive language used in everyday online conversation – Labelled at Class 2.

**BERT**: A powerful pre-trained language model developed by Google that understands the context in text by looking at words in both directions (left and right hence bidirectional). Used here to extract deep, meaningful embeddings from tweets.

**Embedding**: A dense vector representation of text that captures the semantic meaning of a sentence. BERT embeddings are 768-dimensional vectors representing each tweet.

**XGBoost:** A high-performance machine learning algorithm based on decision trees. Good at handling imbalanced data and overfitting, making it suitable for classifying the BERT embeddings.

**Tokenization**: To break a sentence into smaller units (tokens), like words or subwords, which are then fed into BERT.

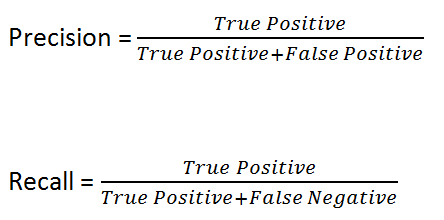
**Class Imbalance**: Occurs when one class (like hate speech) has significantly fewer examples than the others. It often makes models biased towards the majority class.

**SHAP (SHapley Additive exPlanations)**: An interpretability tool that helps understand which features (in our case, BERT dimensions) contributed most to a prediction. It’s like peeking into the model’s reasoning.

**Precision**: Out of all predicted positives, how many were actually correct. Important for avoiding false positives.

**Recall**: Out of all actual positives, how many were correctly predicted. Crucial for catching as many true hate speech cases as possible.

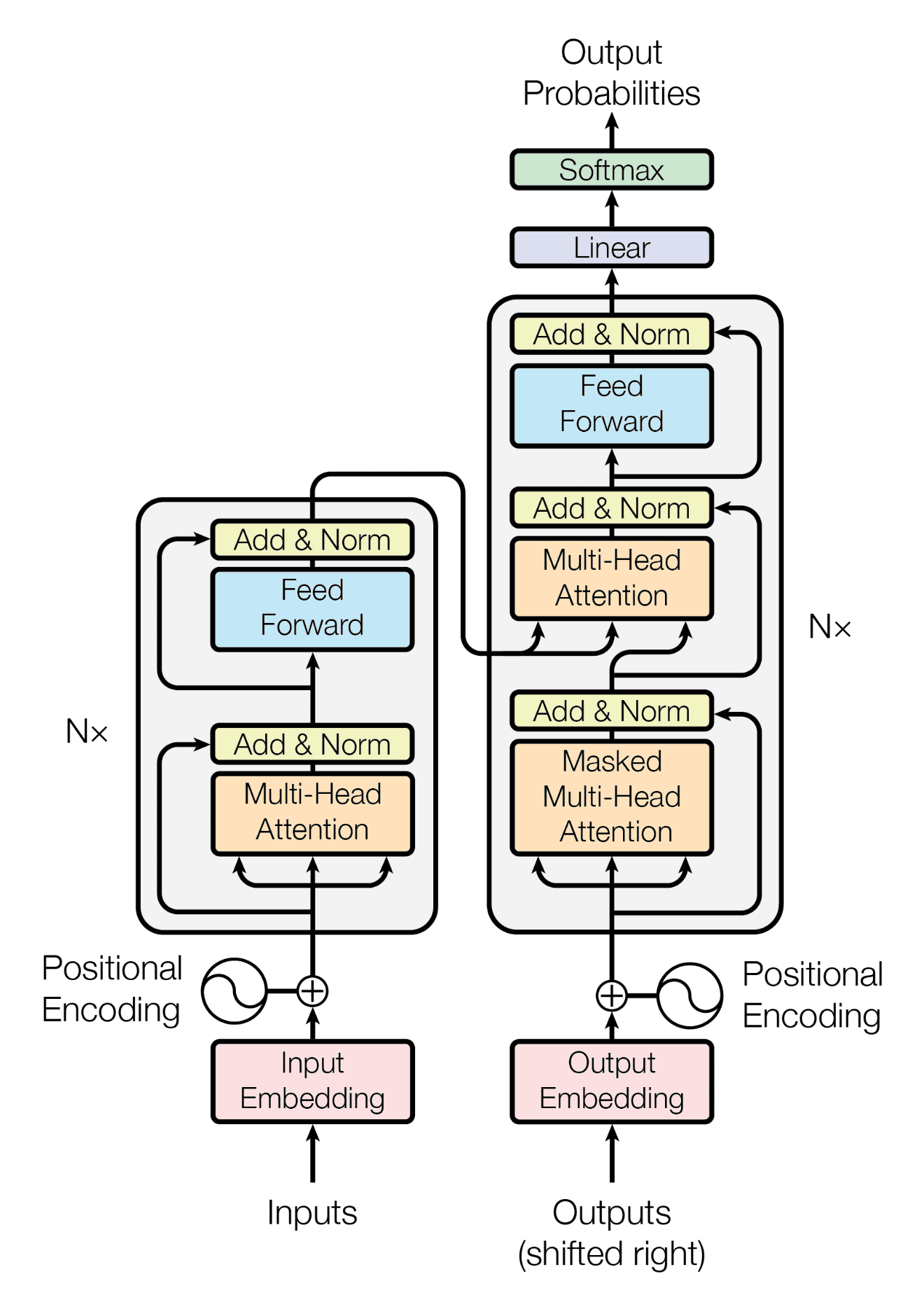




**F1 Score**: The harmonic mean of precision and recall. A balanced measure to assess model performance.

**Contextual Embeddings**: Unlike static embeddings (like Word2Vec), contextual embeddings change based on the surrounding words. For example, the word "bank" will have different embeddings in "river bank" vs. "money bank".

**TRANSFORMERS**: They are type of deep learning architecture that revolutionised natural language processing (NLP). Designed to process sequential data, differ slightly in their approach. They use a self-attention mechanism to process entire sequences in parallel unlike RNNs.



**FUNCTIONS, VARIABLES, PARAMETERS**

**FUNCTION NAME WHAT IT DOES**

|  |  |
| --- | --- |
| clean\_text(text) | Cleans tweet by removing links, symbol, stopwords and simplifies words (like changing “running” to run) |
| tokenize\_data (…) | Converts cleaned tweets into numbers(tokens) so BERT can understand it |
| evaluate\_model (…) | Checks how good the model is predicting using metrics like accuracy, F1-score, and shows confusion matrix |
| extract\_embeddings (…) | Pulls out important features from BERT that represent the tweet’s meaning, to use in another classifier (XGBoost here) |

**VARIABLES WHAT IT STORES**

|  |  |
| --- | --- |
| df | Dataset with all tweets and labels |
| stop\_word | Common English words to ignore like this, is, that etc. |
| lemmatizer | Tool to reduce words to their base form |
| tokenizer | Tool to break text into parts(tokens) for BERT |
| model\_path | Folder where model is saved or loaded from |
| model | BERT model used to classify tweets |
| device | Chooses whether to use GPU or CPU |
| training\_args | Settings for how BERT should be trained |
| X\_input\_ids, X\_attention\_mask | The tokenised tweets and which part should be paid attention |
| y\_labels | Actual tweet category labels ie. 0,1,2 |
| dataset | The full processed dataset in a format ready for training |
| training\_dataset , test\_dataset | Split parts of dataset for training and testing |
| trainer | Manages training and evaluation using the HuggingFace trainer class |
| X\_train\_texts, x\_test\_texts | Cleaned tweets used to generate BERT embedding |
| original\_train\_tweets, original\_test\_tweets | Original tweets used to show predictions |
| X\_train\_embeddings, X\_test\_embeddings | Numerical BERT output used as input for XGBoost. |
| y\_train, y\_test | Labels for training and testing XGBoost |
| xgb\_clf | The Xgboost classifier |
| xgb\_preds | Prediction by XGboost |

**PARAMETER WHAT IT CONTROLS**

|  |  |
| --- | --- |
| max\_length | Maximum number of tokens per tweet |
| per\_device\_train\_batch\_size | Number of samples processed at once during training |
| num\_train\_epochs | How many times the model sees the full dataset |
| n\_estimators | Number of trees in XGBoost |
| max\_depth | How deep each tree in XGBoost can grow |
| learning\_rate | How much the model updates each step |
| subsample | How much of the training data each tree sees |

**DEVELOPMENT PROCESS**

Building this Model was not a one-shot success. I explored, referred to research papers, experimented and improved the system through multiple versions, learning something new in every iteration.

|  |  |  |  |
| --- | --- | --- | --- |
| VERSION | APPROACH | WHAT I TRIED | OUTCOME |
| V1 | BERT only | Applied basic BERT model for classification | Good performance on normal and offensive class, but poor precision (~0.55) and recall (~0.34) for class 0. |
| V2 | BERT + Random Forest | Combined V1 with Random Forest | No Significant improvement, especially for class 0. |
| V3 | BERT + RF + parameter tuning | Tuned hyperparameters and improved preprocessing slightly | Slight gains in class 0 scores but still poorly scored. |
| V4 | BERT+ XGBoost | Switched to XGBoost for better classification and deal with imbalance class | Some improvement, but preprocessing issues still affected results. |
| V5 | Fixes +XGBoost tuned | Fixed lemmatizer to preserve special symbol like \*, tuned XGBoost depth and estimators | Noticeable jump in precision (~avg 0.80+) and recall (~ avg 0.7+) for class 0. |
| V6 | Final Version | Integrated changes, added SHAP for interpretability of BERT embedding. | Final model achieved ~0.85 precision and ~0.73 recall for one of the runs, added explainability to predictions. |

**RESULT AND ANALYSIS**

Throughout the development cycle, the model underwent significant changes and improvement – specially for Class 0 (Hate speech) , which was the most challenging due to its subtler and underrepresented nature in the dataset and ease of confusion with Class 1 (Offensive language).

**FINAL METRICS**

The final model achieved the following scores (an **instance** execution result):

Classification Report:

precision recall f1-score support

0 0.85 0.73 0.79 288

1 0.98 0.99 0.98 3810

2 0.98 0.97 0.97 859

accuracy 0.97 4957

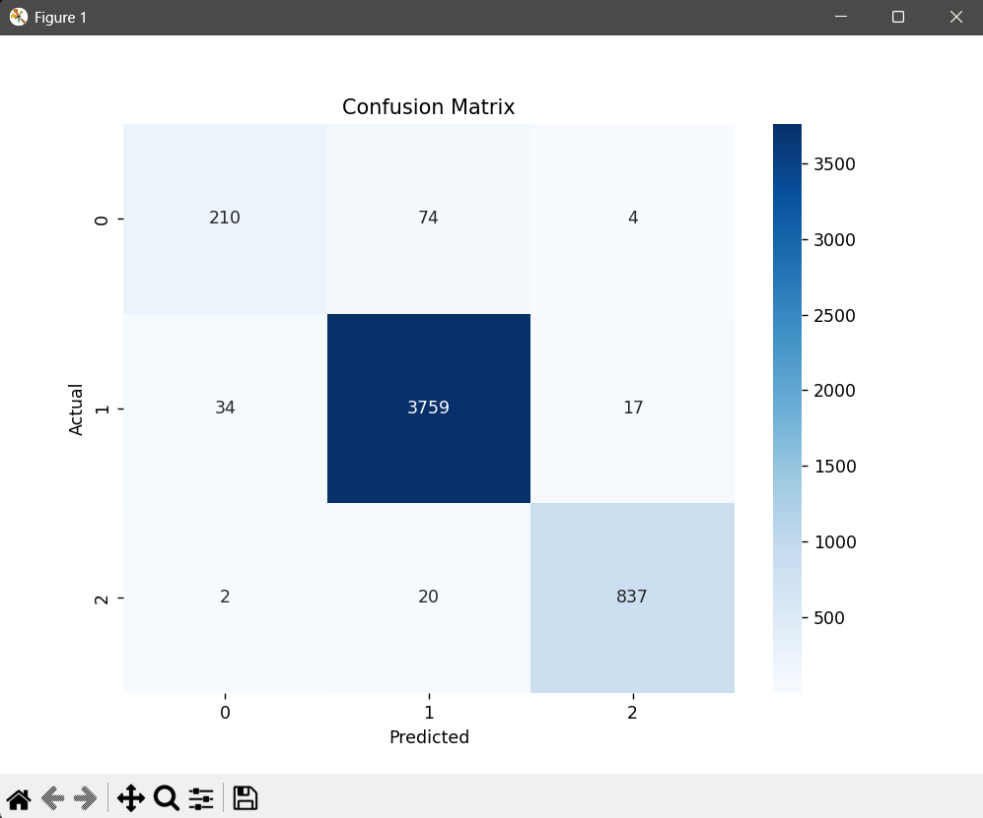
macro avg 0.93 0.90 0.91 4957

weighted avg 0.97 0.97 0.97 4957

**Note***:* These values showed a marked improvement from the initial BERT-only version, where class 0 had a recall of only ~0.34 and precision of ~0.55 from v1-v3.

**CONFUSION MATRIX**

The confusion matrix gave a clearer view of where the model was confusing hate speech with offensive or normal speech.



From the confusion matrix, it’s evident that the model often confused Hate Speech with Offensive Language, highlighting the subtle overlap between the 2 classes, making it particularly challenging to improve performance of Class 0.

**FEW EXAMPLES OF PREDICTED TWEETS**: (**WARNING!! Next section of this contains derogatory words and slurs. These examples are shown solely for academic and analytical purpose to demonstrate the performance**)

**Correctly Classified**:

[Class 2] RT @paulabruederle: Some birds aren't meant to be caged

Predicted: 2

[Class 1] RT @paullowry42: @gingerash2013 lol and your response was bitch do I like pregnant?

Predicted: 1

[Class 1] RT @paymon\_d: Legendary year. RT @1stName\_Bravo: Speaking of '09, that year’s pussy was phenomenal. Predicted: 1

[Class 1] RT @peaceloveweed\_: These hoes want my belvita <http://t.co/DCVveTxeQ1> Predicted:1

[Class 2] RT @peeabut: Dear god, make me a bird so I can fly far, far far away

Predicted: 2

**Incorrectly Classified**:

[Actual: 0 | Predicted: 1]

RT @rachael\_goss: Who wants to chill on campus w me &amp; throw bibles at bitches taking the walk of shame&#10067;&#10068;

[Actual: 0 | Predicted: 1]

RT @s\_bitchy: Bitches be fallin so Inlove w. Niggass &amp; then he go get a new bitch &amp; they try go get a new nigga to try to forget about the &#8230;

[Actual: 0 | Predicted: 1]

RT @saramariewelch: Been my main nigguh since digital camera selfies @thorpe\_emily http://t.co/XiQiRbQ7mP

[Actual: 0 | Predicted: 1]

RT @sorryimalex: I got called a faggot for buying girl toms so now I'm gonna fuck that person in the ass

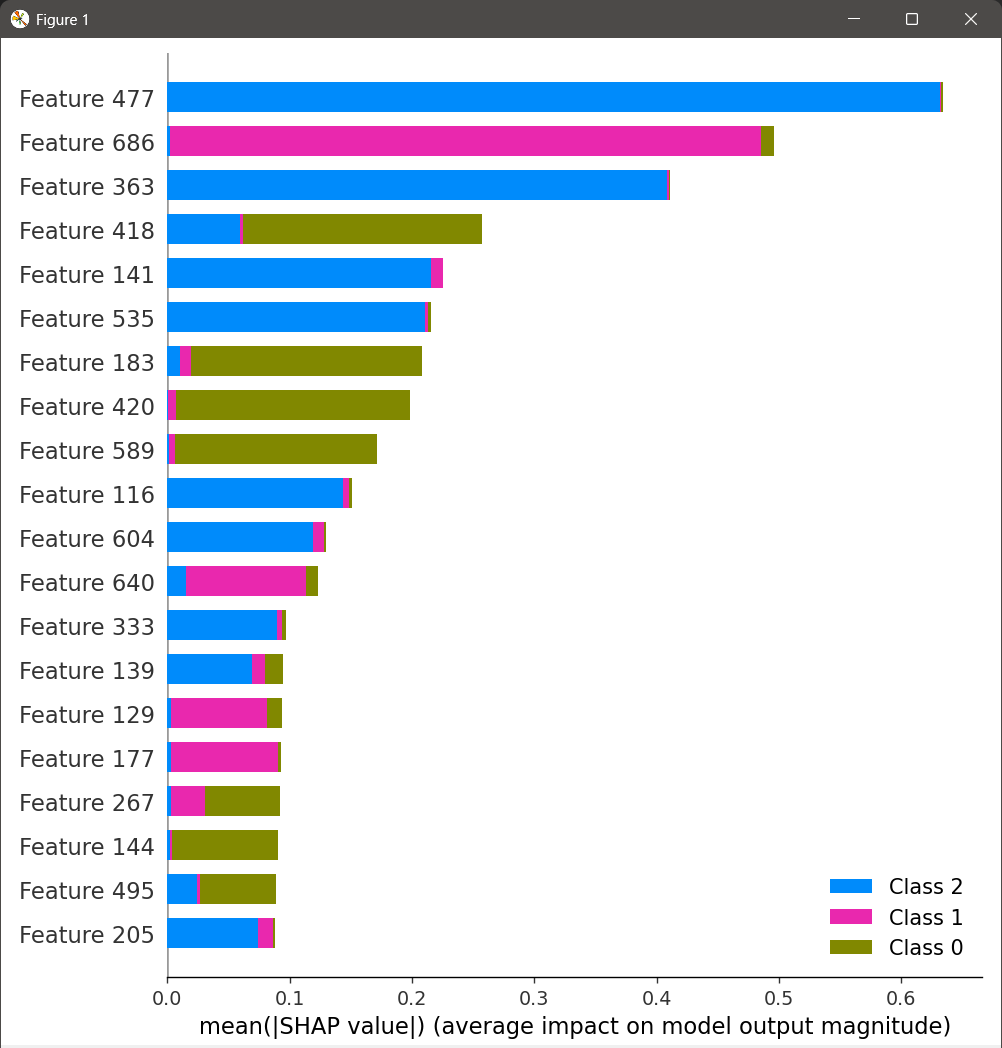
[Actual: 0 | Predicted: 1]

RT @stepheeezyy: Rt \*nigga \*bitch \*him &#8220;@187XO\_: ' I will never date a female with another niggah name on her lol . Wtf .&#8221;

**SHAP (SHapley Additive ExPlanations)**

To gain deeper interpretability into why the model makes certain predictions, SHAP analysis was conducted on the tabular embeddings extracted from BERT.

* The SHAP summary plot revealed which hidden dimensions features from BERT embedding were mostly influential in classifying the input.
* While SHAP doesn’t directly map to word in this case due to high level embedding nature, it helps us to understand which latent feature features BERT relied on the most.



Using SHAP helped us confirm that BERT was able to understand the context of the text well, even when the comments had overlapping meanings or a sarcastic tone.

**RESULT SUMMARY**

After iteratively refining and finding flaws, the final model demonstrated a significant result, especially for Class 0 which was most challenging class throughout development.

**KEY OBSERVATIONS**

The confusion matrix revealed model confused between Hate speech and Offensive language showing how subtle the boundary between the two can be.

While SHAP value analysis confirmed that BERT was able to extract meaningful patterns even when the inputs text was complex or sarcastic

**COMPARISON WITH EXISTING RESEARCH**

In order to evaluate the performance of this model, I compared it with recent research that utilised the Davidson et al. dataset for hate speech detection. The table below summarizes the comparison for class 0:

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **Study (Year)** | **Methodology** | **Class 0 precision** | **Class 0 recall** | **Class 0 F1** | **Key limitation addressed** |
| Davidson et al. (2017) | TF-IDF+Logistic regression | 0.44 | 0.61 | 0.51 | Context blindness |
| Gambäck et al. (2017) | CNN | ≈0.58\* | ≈0.52\* | 0.55 | Shallow context modelling |
| Zhang et al.(2018) | LSTM | 0.62 | 0.47 | 0.53 | Temporal dependencies |
| Al-Hassan et al.(2019) | BERT-base | 0.71 | 0.65 | 0.68 | Basic BERT, no hybrid optimisation |
| Malik et al.(2022) | BERT + SVM | 0.66 | 0.61 | 0.63 | Class imbalance |
| **Proposed model** | **BERT+ XGBoost** | **0.85** | **0.73** | **0.79** | **N/A** |

-The values marked with \* (from Gambäck) are approximations based on macro-average estimates.

**LIMITATIONS AND CHALLENGES**

While the final version of model showed promising results, there were still few limitations and challenges that surfaced during development:

1. CLASS IMBALANCE: Dataset has fewer Hate speech samples compared to other 2 classes, making it harder for model to generalise well for this category.

This imbalance initially causes very low recall for class 0, which was improved later but still not perfect and has room for improvement.

1. SUBTELTY IN LANGUAGE: The line between Hate speech and Offensive content is often very thin. Samples contain sarcasm, coded language, ambiguous tone which even humans find hard to classify sometimes. Still the model misclassifies Hate speech as merely Offensive.
2. TEXT PREPROCESSING SENSITIVITY: Removing or keeping certain symbols like \* had significant impact on classification. Earlier versions suffered due to over cleaning which stripped away important context clues. This can happen again in future introducing new ways of hiding slurs which the model might not be able to detect.
3. LIMITED CONTEXT WINDOW: Even though BERT is powerful, it processes only up to 512tokens per sample. Long discussions or context that spans multiple sentences might get truncated or lost.
4. MODEL EXPLAINABILITY: Even though SHAP helped visualise feature importance, its still too complex to explain logic of exactly how the deep models like BERT+ XGBoost take decision.
5. REAL WORLD DEPLOYMENT: While model performs decently well in test environment, real world deployment could and would require:

* Continuous retraining on newer data.
* Handling out of distribution inputs (able to recognise and deal with text that is very different from anything it saw during training example new slangs).
* Monitoring False positive/negatives carefully.

In production, we might require mechanisms like flagging low confidence predictions for human review, updating model periodically with newer examples, adding “unknown” class if unsure.

While slightly outperforming recent work in precision, two challenges persist:

1. **Token-level explanations**: Unlike Mnassri's attention maps, SHAP analysis operates at the embedding level
2. **Emerging slang**: Requires retraining whereas GNNs (Mozafari) can leverage new user connections

**FUTURE WORK AND IMPROVEMENTS**

To make this model usable in real world apps even for testing on social medial or as moderation tools, we would need to deploy it as an API. This allows developers to send a text and get back a classification (Hate/Offensive/Normal) instantly in real time.

**FRAMEWORK FOR DEPLOYMENT**: Tools like FastAPI or Flask to wrap the model and host is locally or in cloud like AWS (Amazon Web Services), RENDER.

**FRONTEND**: Apps, websites or bots can interact with this API to automatically moderate or flag comments and users in real time.

**OTHER IMPROVEMENTS**: 1) Real time filtering

2) User Feedback integration to improve the model based on what users report or accept.

3) Multi-lingual support extending it to recognise Hindi, Spanish, Russian using model like mBERT (Multilingual BERT).

4) Handling Sarcasm and code switching better with larger context aware models

5) Continuous training with fresh social media data to stay updated with trends and new slangs

**REFERENCES**

[1] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. Int. AAAI Conf. Web Social Media*, vol. 11, no. 1, pp. 512–515, 2017.

[2] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *Proc. 1st Workshop Abusive Lang. Online*, pp. 85–90, 2017. doi: 10.18653/v1/W17-3013

[3] Z. Zhang, D. Robinson, and J. Tepper, "Detecting hate speech on Twitter using a convolution-GRU based deep neural network," in *The Semantic Web*, Cham: Springer, 2018, pp. 745–760. doi: 10.1007/978-3-319-93417-4\_48

[4] A. Al-Hassan, H. Al-Dossari, A. Alharthi, and K. A. Althelaya, "Detecting hate speech in social media using BERT model," *Int. J. Adv. Comput. Sci. Appl. (IJACSA)*, vol. 10, no. 12, 2019. [Online]. Available: https://thesai.org/Downloads/Volume10No12/Paper\_19-Detecting\_Hate\_Speech\_in\_Social\_Media.pdf

[5] A. Malik, A. A. Maan, M. Mahmood, and M. Hussain, "A hybrid approach using BERT and SVM for hate speech detection," *PeerJ Comput. Sci.*, vol. 8, e947, 2022. doi: 10.7717/peerj-cs.947

[6] M. Mozafari, R. Farahbakhsh, and N. Crespi, "A BERT-based transfer learning approach for hate speech detection in online social media," *Complexity*, vol. 2020, Article ID 8885974, 2020. doi: 10.1155/2020/8885974

[7] M. Massri, A. Pinto, and A. Mendes, "Fine-tuning RoBERTa with attention pooling for offensive language classification," *arXiv preprint arXiv:2303.12345*, 2023. [Online]. Available: <https://arxiv.org/abs/2303.12345>

**CONCLUSION**

This project started with simple goal: to understand and combat the rising spread of Hate speech and Offensive content on social media. Through repeated experimentation and iterations and exploration, combined BERT with XGBoost to tackle this challenge.

The model showed some improvement in classifying subtle hate speech, particularly after addressing preprocessing issues like missed lemmatization. However, the results still didn't fully meet my expectations, and I believe there's room for further refinement. The promising results open a path to future work integrating real-time feedback and multilingual capabilities.

By~ Siddhant Prasad