Enhanced Profanity Obfuscation Using Toxic Comment Classification

Chandish Kumar Reddy Somu
George Mason University
csomu@gmu.edu

Aryan Srivatsava Dande
George Mason University
adande@gmu.edu

Sai Kiran Reddy Mothe
George Mason University
smothe@gmu.edu

1 Introduction

1.1 Task / Research Question Description

There have been many complaints about the toxic language usage by users in many social media platforms. For example, Instagram is one of the most widely used application across the globe. It gives access to speak out or publicize whatever users want to. There are many cases where a user post something and receive negative responses through comment section. Posts related to religion, politics, and other sensitive topics might lead to generate different personal opinions in comment section resulting in abusive language. This might negatively impact and influence other users (mainly who are under 18). So, our team designed an algorithm that has the potential to tackle this issue. It involves developing methods to replace or disguise offensive language, typically through the use of substitutions, filters, or other forms of modification. The aim is to mitigate the impact of explicit or inappropriate content in communication channels, online platforms, or any context where maintaining a respectful or inclusive environment is crucial. We explore various techniques, including natural language processing algorithms, rule-based systems, and machine learning approaches, to identify and replace profanity with alternative, less objectionable content.

1.2 Motivation and Limitations of existing work

Existing research in the realm of profanity obfuscation, exemplified by a selected paper in ACL research, has primarily focused on partially masking offensive words, including those containing adult language or explicit content. However, these approaches exhibit limitations, as highlighted by the example in the provided response: the failure to recognize certain abusive words, leading to incomplete masking. For instance, the substitution of "fuck" with "f*ck" is a common approach, but it may overlook variations or miss instances where explicit words are not entirely replaced, such as the case of "porn" being left unaltered.

Moreover, some implementations may demonstrate greater efficiency in identifying profanity within entire sentences. However, the existing literature lacks a comprehensive solution for achieving more accurate word masking, addressing the shortcomings identified in the aforementioned paper. The specific focus of our project idea revolves around enhancing the precision of profanity word masking. We aim to overcome the challenges posed by the limitations in the prior work by developing a more sophisticated and accurate approach to recognize and replace offensive words.

In summary, while existing research has delved into profanity obfuscation, our project distinguishes itself by honing in on the accuracy of word masking, aiming to address and rectify the limitations observed in the selected ACL research paper. This novel approach seeks to contribute to the refinement of profanity-filtering mechanisms, enhancing their effectiveness and minimizing instances of overlooked or incompletely masked offensive language.

1.3 Proposed Approach

The existing code implementation, as outlined in the research paper "The State of Profanity Obfuscation in Natural Language Processing Scientific Publications," primarily focuses on masking specific bad words like "f*ck" to maintain formality in the content. One significant limitation of the current approach is its reliance on a predefined dataset of bad words. This means

that it may struggle to detect and handle profanity words not present in its training data, limiting its adaptability to emerging or context-specific offensive language. To address these issues and enhance the efficiency of profanity detection, our team is developing an improved version of the existing approach. In our proposed approach, we leverage a pretrained model trained on a substantial dataset of 200,000 human-labeled samples. This pretrained model is expected to exhibit a broader understanding of profanity, enabling it to detect a wider range of offensive language, including words not explicitly listed in a predefined bad words dataset.

Focusing on highly profane language is essential for applications such as content moderation, where the goal is to filter out offensive content and maintain a safer online environment. By concentrating on sentences labeled as ”severe-toxic” or ”obscene,” the model is trained to prioritize the identification of the most egregious forms of offensive language, providing a more effective solution for the intended use case.

Methodology:

1. Dataset Selection: The dataset comprises diverse textual features, with a specific emphasis on the ’comment text’ column. The focus is directed towards sentences containing highly profane language, which are identified through the ”severe-toxic” or ”obscene” labels in the dataset.

2. Labeling Strategy: Sentences falling into either the ”severe-toxic” or ”obscene” categories are labeled as 1, indicating the presence of highly offensive content. Sentences not falling into these categories are labeled as 0, denoting the absence of highly profane language.

3. Model Training: The machine learning model is trained on this refined dataset, emphasizing the binary classification task of identifying sentences with highly profane language.

4. Performance Evaluation: Model performance is assessed based on its ability to accurately classify sentences as highly profane or not. Evaluation metrics such as precision, recall, and F1 score are employed to gauge the model’s effectiveness in handling offensive content.

Additionally, our approach incorporates advanced linguistic techniques, such as stemming and lemmatization, to process input sentences. This preprocessing step aims to standardize word forms, reducing the complexity of profanity detection and ensuring that variations of offensive words are appropriately identified. A key innovation in our approach involves calculating probabilities of profanity for each word in the input sentence. This nuanced approach allows for a more context-aware profanity detection mechanism, assigning probabilities based on the likelihood of a word being offensive in a given context.

The proposed approach aims to tailor the model’s focus to the identification of highly profane language by narrowing down the classification task to sentences falling under the ”severe-toxic” or ”obscene” categories. This targeted labeling strategy optimizes the model for content moderation purposes, aligning with the specific needs of the project. Focusing on highly profane language is essential for applications such as content moderation, where the goal is to filter out offensive content and maintain a safer online environment. By concentrating on sentences labeled as ”severe-toxic” or ”obscene,” the model is trained to prioritize the identification of the most egregious forms of offensive language, providing a more effective solution for the intended use case.

Finally, the output of our approach involves masking the profane words in the original sentence. This ensures that the formal tone and context of the content are maintained while effectively addressing the limitations of the existing code implementation. In summary, our proposed approach seeks to overcome the limitations of the existing

method by incorporating a pretrained model, linguistic preprocessing techniques, and probabilistic profanity detection, ultimately aiming for a more robust and adaptable profanity obfuscation solution.

1.4 Likely challenges and mitigations

The primary challenge in tackling the profanity obfuscation task lies in the sensitivity of the data to various factors, including geographical and language-specific nuances. Profanity is often culturally and regionally dependent, with certain words being considered offensive in one context but acceptable in another. Additionally, the presence of ambiguous words further complicates the task, as a term that may be perceived as abusive in specific regions might be entirely benign in others.

To address these challenges, our approach involves preparing and tuning the model to account for these geographical and language-specific factors. This includes incorporating region-specific datasets or adapting the model to recognize and handle variations in the perception of profanity across different linguistic and cultural contexts. However, even with meticulous preparation, it's challenging to anticipate all possible linguistic subtleties and variations. Contingency plans are crucial in the face of unexpected difficulties or if experiments do not proceed as planned. Some potential contingency measures includes: Embrace an iterative development process, allowing for ongoing adjustments based on feedback and observed performance. This ensures a dynamic approach that adapts to emerging challenges. Implement mechanisms for the model to dynamically adapt its profanity recognition based on regional or cultural variations. This adaptability can enhance the model's effectiveness across diverse contexts. Define and utilize robust evaluation metrics that capture not only accuracy but also account for false positives and false negatives. This allows for a more comprehensive assessment of the model's performance.

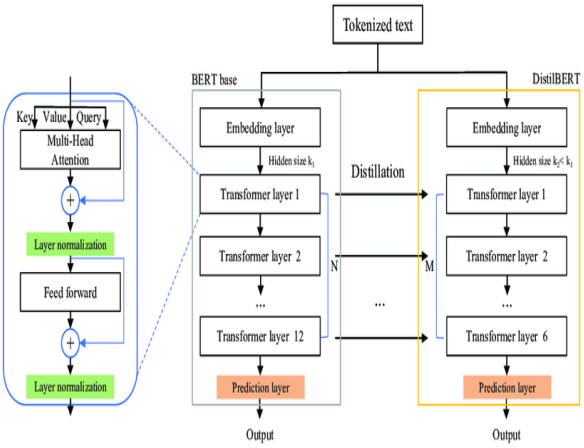
In summary, the task is tough because it depends on cultural and regional differences, making it important to be proactive and adaptable. We have plans in place, like keeping a close eye on how well the system works, making regular improvements, and adjusting to unexpected challenges. These steps are crucial to making sure our profanity obfuscation system is strong and effective, even when faced with unexpected problems.

2 Approach

2.1 Distil-Bert-Based-Model

"DistilBERT" is a variant of the BERT (Bidirectional Encoder Representations from Transformers) model that has been distilled for efficiency while retaining much of its performance. "Distil" stands for distilled, indicating that the model has undergone a compression or distillation process to reduce its size and computational requirements. In summary, DistilBERT is a more streamlined version of the BERT model, achieved through knowledge distillation. It maintains the transformer architecture and tokenization principles while being more computationally efficient, making it a popular choice for various NLP tasks.

2.2 Why DistilBERT?



It is distilled or compressed from BERT, resulting in a smaller model with fewer parameters. This makes DistilBERT more computationally efficient and faster to train and deploy. The smaller size of DistilBERT makes it more suitable for applications with limited computational resources or deployment on devices with constraints. Generally faster to train due to its smaller size.

2.3 Model Architecture And Design

The term "Distil" in DistilBERT refers to the distillation process. Distillation involves training a smaller model (DistilBERT) to replicate the behavior of a larger model (BERT). The distilled model is designed to capture essential knowledge from the larger model while being more computationally efficient. Like BERT, DistilBERT

is based on the transformer architecture, which includes attention mechanisms for capturing contextual relationships between words in a sequence. The DistilBERT model architecture and design follow a distilled approach, aiming to maintain performance while reducing the size and computational requirements compared to the original BERT model. Here are the key aspects of the DistilBERT architecture:

Tokenization and Embeddings, Model Size Reduction, Layer Pruning, Knowledge Distillation Loss, Parameter Sharing, Pre-training and Fine-tuning.

2.4 Design Setup

checkpoint 1: Load the dataset(list of profane words). Reproduce the results. Contains a function that checks if a given input word is stored in its dataset and returns the obfuscated input word (masked word) only if that input word is present in its dataset.

Uses a pre-trained a distilbert-based-model on "jigsaw-toxit-comment-classification" dataset. Fine-tunes the model by adding the data used in checkpoint 1 to validation set. (Adding profane words to the validation set can help model to find optimal parameters such that profane words are identified) Sets the threshold too high for classification which helps the model to prevent detecting false positives.(predicts label 1 only if the sentence if severely toxic or obscene) Checks for each word in a given sentence and classifies if it resembles profanity. All the words that are classified as profane are masked/obfuscated.

3 Experiments

3.1 Datasets

We used Jigsaw MultilingualToxic Comment Classification Datasets which is available kaggle. https://www.kaggle.com/c/jigsaw-multilingual-toxic-comment-classification/data see for instructions. In each provided file, the primary dataset is found in the comment_text column. This column contains the text of comments classified as either toxic or non-toxic, indicated by values of 0 or 1 in the toxic column. The comments in the train set are exclusively in English and are sourced from either Civil Comments or Wikipedia talk page edits. In contrast, the test data's comment_text columns include content in multiple non-English languages. The *-seqlen128.csv files encompass training, validation, and test data processed for input into BERT, a language model.
The another dataset we have taken is jigsaw multilingual comment(Google api) from kaggle.https://www.kaggle.com/datasets/miklgr500/jigsaw-train-multilingual-coments-google-api/data.This datasets have multiple languages that are translated using google api.

3.2 Data Preprocessing and Implementation

3.2.1 Implementation

Github link:
https://github.com/ARYANSRIVATSAVA/CS678checkpoint2
Github link for Checkpoint 2.

3.2.2 Data Preprocessing

We have done datapreprocessing for our model.When it comes to getting raw text data ready for machine learning models, data pre-processing is essential. Special characters and noise can all have a big effect on how well a model performs in toxicity prediction tasks. Through a number of data cleansing methods used in the pre-processing stage, this study tackles these issues.
1.Removing Punctuation:To guarantee uniformity and ease further analysis, a collection of punctuation marks and special characters were found and eliminated from the dataset.
2.Text cleaning:The clean function eliminates newline characters, user mentions, IP addresses, and hyperlinks from text in a general text cleaning manner.
3.Contraction Handling:The handle_contractions method tokenizes the text to address contractions and guarantee a consistent word representation.
4.Quote Fixing:The fix_quote function fixes typos in quotes by removing apostrophes from words that start with them.The function goes over the tokens iteratively, correcting quotations as needed to improve the text's overall consistency.
5.Mispelling Correction:The correction of typical misspellings in the dataset is handled by the combined use of the get_mispell and replace_typical_misspell methods.

3.2.3 Undersampling

Under-sampling involves randomly removing instances from the majority class until a more balanced distribution is achieved. In the context of this research, the severe toxic class (label 1) was under-sampled to match the number of instances in the non-toxic class (label 0). This balanced dataset was then used for training and evaluating toxicity prediction models.

3.3 Metrics

We use precision, recall, F1 score to evaluate our model against existing model.

3.3.1 Epochs

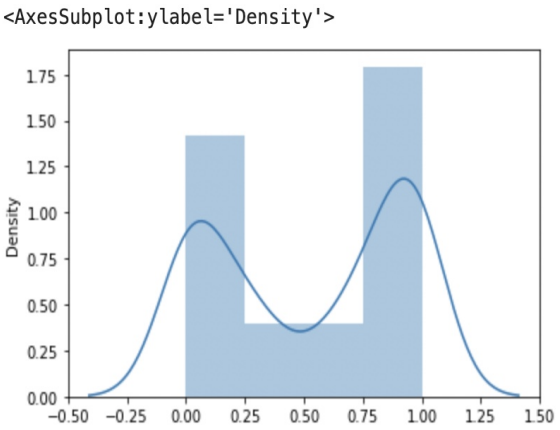
The chosen number of epochs dictates how many times the entire training dataset is processed during model training.
To balance training effectiveness and computational resources, a cautious decision of 4 epochs was made, guaranteeing convergence and avoiding overfitting.

3.3.2 Batchsize

The batch size defines the number of training samples utilized in one iteration of model training. It significantly influences training speed and memory requirements.A batch size of 16 times the number of replicas in sync (to leverage distributed training) was chosen to optimize GPU memory usage and parallelize training across replicas efficiently.

3.3.3 Optimizing Model Thresholds for Enhanced Precision

Choosing an appropriate classification threshold is pivotal for achieving optimal precision in machine learning models. This study focuses on the gradient analysis of correct predictions across consecutive thresholds, identifying a significant change in behavior within the range of 0.7 to 0.8. Leveraging the elbow method, the research aims to pinpoint an optimal threshold that maximizes the model's precision for further predictions.



The above diagram explains the distribution of prediction probabilities for test dataset in range(0,1).

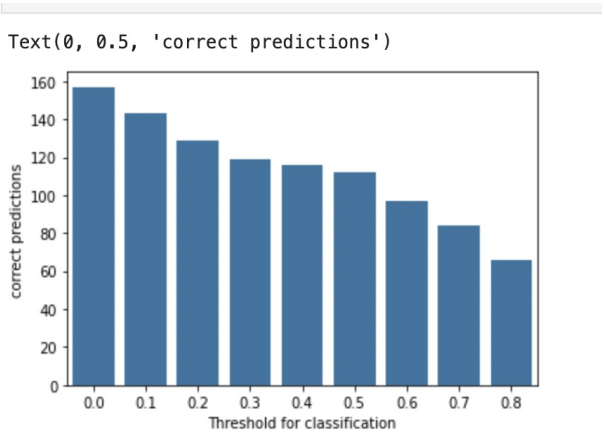
Gradient Analysis:

1.Gradient Calculation:

- Description: Gradients of correct predictions between consecutive thresholds are computed to assess the model's performance sensitivity to threshold variations.
- Implementation: The differences in correct predictions are calculated for each threshold pair, forming a gradient series.

2. Identifying the Elbow:

- Description: The elbow method is applied to detect a point of inflection in the gradient series, indicative of a significant change in behavior.
- Implementation: By analyzing the gradients, the point where the rate of change exhibits a pronounced shift is identified as the elbow.



3. Threshold Selection:

- Description: The threshold corresponding to the elbow point is selected as the optimal classification threshold for further model predictions.
- Implementation: The threshold at which the elbow occurs is determined and set at 0.75 based on the observed inflection point.

The observed inclination in gradients between thresholds 0.7 and 0.8 signals a critical region in which the model's performance is particularly sensitive to threshold variations. The elbow method, a recognized technique for identifying optimal points in a dataset, successfully pinpoints the threshold of 0.75 as the point of inflection. This threshold is subsequently adopted for further model predictions.

3.4 Robustness

Our research employs Three tests for Robustness i.e MFT(Minimum Functionality Test),Invariance Testing and directional Expectation Testing.

MFT: Our study evaluates a machine learning model's predictive power using the Minimum Functionality Test (MFT) technique. The checklist.editor function is employed in the study to create a wide range of test cases. The positive terms are used to assess the model's capacity to identify desired results, while the profane words ('fuck,' 'ass,' and'suck') are used as negative instances to test the model's resistance to unsuitable content. When the accuracy of the model is assessed using the resulting dataset, which consists of words that emulate both profane and realistic situations, a remarkable 100% success rate is found.

```
#MFT test summary
test.summary()
```

```
Test cases:      28
Fails (rate):    0 (0.0%)
```

Invariance Testing: Our study uses Invariance Testing (IVT), a methodology created to evaluate the model's performance under multiple perturbations, to investigate a machine learning model's robustness. Using perturbation functions (i.e., change_professions(), change_names(), change_location(), perturb_typos(), and perturb_contractions()) to modify the input data, the research assesses the model's ability to retain accuracy in the face of various linguistic alterations. The final dataset, produced using IVT techniques, shows a 75% accuracy rate, suggesting possible difficulties in precisely predicting sentences that contain names.

```
#IVT TEST
test2.summary()
```

```
Test cases:      4
Fails (rate):    3 (75.0%)

Example fails:
1.0 Aaron had fun traveling to Mexico
0.0 Aaron had fun traveling to Saudi Arabia
0.0 Aaron had fun traveling to Algeria

----
1.0 Michael had fun traveling to Mexico
0.0 Michael had fun traveling to Saudi Arabia
0.0 Michael had fun traveling to Algeria

----
1.0 Michael had fun traveling to Mexico
0.0 Michael had fun traveling to Peru

----
```

Directional Expectational Testing: Directional Expectation Tests (DIR) are used in this study to examine a machine learning model's predictive capacity. In particular, the study introduces directional variations in the input data by utilising the perturbation function perturb.add_negation(). The final dataset—produced using DIR techniques—shows an astounding 0 failrate. A closer look reveals a possible problem with name-containing sentence prediction, though. In order to highlight the model's advantages and shortcomings, this paper describes the DIR tests' methodology, approach, findings, and implications.

```
#DIR TEST
test3.summary()
```

```
Test cases:      63
After filtering: 29 (46.0%)
Fails (rate):    0 (0.0%)
```

3.5 Model Behaviour

Scenario 1: "He is not ass" (Toxic Calculation: 87_): The model gives this sentence a significant toxicity score even though it doesn't contain an explicit profanity term. This implies that the model's toxicity prediction takes into account the sentence's overall meaning and is sensitive to contextual cues.
In the second scenario, "He is an ass" (Toxic Calculation: 97%): On the other hand, the model gives a higher toxicity score in the case of an explicit profanity term. This suggests that the model reacts with a more pronounced toxicity prediction after correctly identifying the profanity's severity.

3.6 Multilinguality

The "Jigsaw-toxic-comment-classification-challenge" dataset comprises English sentences containing profanity, toxic, and obscene words. To expand its applicability to multilingual scenarios, the dataset has been translated into French, Spanish, Italian, and Turkish. Implications: During translation, profane words in English may be substituted with culturally equivalent non-profane terms in the target language. This linguistic variation poses a challenge for the model, as it is trained on English sentences containing profanity but lacks similar instances in other languages.
The trained model demonstrates robust proficiency in detecting profanity words in English, where it has been exposed to a diverse set of sentences containing explicit language.The model's robustness in English is attributed to the abundance of training examples with profanity words. However, this proficiency does not necessarily extend to other languages due to the aforementioned translation-induced substitutions.

Lack of Representative Data: The translation process results in sentences where profane words are replaced by non-profane words that convey similar meanings in other languages. Consequently, the model is deprived of representative examples of profanity in its training data for non-English languages.The model struggles to generalize profanity detection to languages other than English, as it lacks exposure to sentences with translated profanity.This research underscores the challenges posed by translations in multilingual profanity detection, emphasizing the need for targeted strategies to overcome these limitations. The insights gained from this study contribute to ongoing efforts in refining models for cross-linguistic applications, ensuring more accurate and culturally sensitive profanity detection in diverse linguistic contexts.

3.6.1 Outputs(multilingual)

input sentence1: este usuario ni siquiera llega al rango de hereje por lo tanto debería ser quemado en la barbacoa para purificar su alma y nuestro aparato digestivo mediante su ingestión skipe linkin 22px honor valor leltad 1748 13 mar 2008 utc

translated sentence: This user does not even rank heretic. therefore it should be burned at the barbecue to purify his soul and our digestive tract by ingestion. skipe linkin 22px honor, courage, leltad. 13 Mar 2008 17:48 (UTC)

profane words in a sentence:
{'siqueira', 'barbacoa', 'debera', 'su'}

input sentence2: in riguardo a cio che ho pubblicato e tu hai gi tolto dai maroni lo trovo ingiusto e un po comunista sei pregato di non scassarmi il cazzo e te lo chiedo gentilemnte

translated sentence: in regard to what I posted and you've already removed from the Maronites. I find it unfair and a little Communist. please do not scassarmi cock, and I ask gentilemnte
profane words in a sentence:
{'cio', 'sassari', 'il'}

input sentence3: razza di idioti vi siete bevuti il cervello cos questa intromissione le vostre beghe personali risolvetele nelle vostre pagine personali in più io ho votato per la cancellazione quindi andate a lo

translated sentence: breed of idiots !!! you have drunk the brain ???? so is this meddling ????? your personal squabbles risolvetele in your pages (personal)! in addition, I have voted for the cancellation !!! so go ..... it!
profane words in a sentence:
{'vostre', 'beruti', 'idioti', 'le', 'il'}

3.7 Results

Provide a table comparing your results to the published results.

First table displays the results of our approach.

```
from sklearn.metrics import classification_report
print(classification_report(test_output,y_test))
#Accuracy scores - English language only
```

	precision	recall	f1-score	support
0	0.98	0.84	0.91	7789
1	0.82	0.97	0.89	5673
accuracy			0.90	13462
macro avg	0.90	0.91	0.90	13462
weighted avg	0.91	0.90	0.90	13462

Second table displays the top 3 existing approaches for the same research idea.
see https://github.com/dimitrismistriotis/alt-profanity-check/blob/master/README.md for instructions.

PACKAGE	TEST ACCURACY	BALANCED TEST ACCURACY	PRECISION	RECALL	F1 SCORE
PROFANITY-CHECK	95%	93%	86.1%	89.6%	0.88
PROFANITY-FILTER	91.8%	83.6%	85.4%	70.2%	0.77
PROFANITY	85.6%	65.1%	91.7%	30.8%	0.46

3.8 Resources

We have used 2 ORC accounts using TPU.
8GB Memory/core, Time: 20 hours per session
Cores: 8.
Total time spent: 150-200 hours.

3.9 Discussion

While the results of our approach align with those of existing methodologies, the impact of advanced data pre-processing techniques is discernible. The incorporation of advanced spelling corrections and dictionary-based contraction handling significantly contributes to the robustness of the trained model. Furthermore, the application of random seed variation highlights the model's stability, ensuring consistent prediction accuracies across different initializations

3.10 Error Analysis

input: This stupidd= mothrfuckr has come here

result: This stupid*= moth*fuckr has come here

input: This stupidd= mothrfuckrrr has come here
result: This stupi*d= mothrfuckrrr has come here

The model struggles to accurately detect misspelled profane words when the deviations exceed a certain threshold.
Example: In the input 'This stupidd= mothrfuckrrrr has come here,' the model corrects 'stupidd=' to 'stupid*=' and 'mothrfuckrrrr' to 'mothfuckr,' potentially overlooking the intended profanity.

NOTE: Error analysis for non-english languages explains that the translated dataset doesn't provide enough information for the model to correctly detect words resembling profanity.

4 Related Work

In recent times there is a rapid growth of profane words usage over social media platforms.In the work done by rohithdas https://github.com/therohitdas/profanityjs. The ProfanityJs project focuses on efficient and straightforward profanity filtering in text, addressing four key aspects. Firstly, it employs a trie data structure for storing profanity words, ensuring fast and effective profanity detection for applications like chat filtering and social media moderation. Additionally, the library supports homoglyph search to identify offensive words written with visually similar characters, enhancing its ability to catch creatively encoded profanity. Moreover, ProfanityJs offers the flexibility to mask profanity with user-defined characters, providing a solution for scenarios where censorship is preferred over deletion. Overall, it is a lightweight and customizable library suitable for integration into diverse applications.

https://www.kaggle.com/code/mobassir/understanding-cross-lingual-models.This kaggle kernel explains the understanding and findings of cross lingual models. https://www.kaggle.com/code/xhlulu/jigsaw-tpu-distilbert-with-huggingface-and-keras This kaggle notebook explains how to create basic distil-bert based multilingual model training environment usin tpu.

5 Conclusion

The research introduces a strategic methodology for labeling dataset, with a specific focus on identifying highly profane language to enhance content moderation efforts. While acknowledging the importance of addressing misspelled profane words to ensure real-world effectiveness, the model demonstrates robust overall performance. Its tailored design aligns with the project's objectives, contributing to the development of content moderation systems that are not only focused but also efficient. The model's high robustness on real-world data suggests its capability to effectively handle various forms of profanity obfuscation, reinforcing its potential for deployment in practical scenarios where dealing with creatively disguised profanity is a common challenge. Despite, not being able to interpret some of the contextual intentions in sentences with unusual spelling formats(refer error analysis), our model is extremely robust on identifying profanity in real world data.

References

Dimitrios Mistriotis https://github.com/dimitrismistriotis/alt-profanity-check/blob/master/README.md

MOBASSIR

https://www.kaggle.com/code/mobassir/understanding-cross-lingual-models/notebook

XHLULU

https://www.kaggle.com/code/xhlulu/jigsaw-tpu-distilbert-with-huggingface-and-keras

https://link.springer.com/article/10.1007/s10579-022-09582-8

http://figures-of-speech.com/2016/01/proxy-words.htm

https://paperswithcode.com/paper/rephrasing-profanity-in-chinese-text

Profanity as a Self-Defense Mechanism and an Outlet for Emotional Catharsis in Stress, Anxiety, and Depression Waqar Husain , Samia Wasif , and Insha Fatima
https://downloads.hindawi.com/journals/drt/2023/8821517.pdf?_gl=1*s1ntdc*_ga*MjA3NDUyMDI0Ni4xNzAxNzQ4NDk5*_ga_NF5QFMJT5V*MTcwMTc0ODQ5OS4xLjAuMTcwMTc0ODQ5OS42MC4wLjA.&_ga=2.151824441.1022049569.1701748499-2074520246.1701748499

The sound of swearing: Are there universal patterns in profanity? Shiri Lev-Ari1 · Ryan McKay
https://link.springer.com/article/10.3758/s13423-022-02202-0#citeas

Rohitdas https://github.com/therohitdas/profanityjs

https://web.stanford.edu/class/archive/cs/cs224n/cs224n.1184/reports/6837517.pdf

A Survey of Toxic Comment Classification Methods Kehan Wang, Jiaxi Yang, Hongjun Wu
https://arxiv.org/pdf/2112.06412.pdf

Challenges for Toxic Comment Classification: An In-Depth Error Analysis Betty van Aken1 , Julian Risch2 , Ralf Krestel2 , and Alexander Loser
https://aclanthology.org/W18-5105.pdf

6 Appendix: Contributions

6.1 Aryan Srivatsava Dande

Written code to preprocess data, train and tune the model.

6.2 Kiran Reddy Mothe

Performed error analysis and added functions containing data-output modifications

6.3 Chandish Kumar Reddy Somu

Implemented functions for robustness and multilinguality.

7 Screenshots

https://github.com/ARYANSRIVATSAVA/CS678checkpoint2/blob/main/outputs