CS5246 Project (Team 4 - RedactOps) – No Regrets

Ayush Goyal (A0274831B)
School of Computing
National University of Singapore
Singapore, Singapore
e1124577@u.nus.edu

Jasjeet Singh Siddhu (A0274722A)
School of Computing
National University of Singapore
Singapore, Singapore
e1124468@u.nus.edu

Shamik Banerjee (A0276524Y)
School of Computing
National University of Singapore
Singapore, Singapore
e1132257@u.nus.edu

Vishwanath Dattatreya Doddamani
(A0286188L)
School of Computing
National University of Singapore
Singapore, Singapore
e1237250@u.nus.edu

Abstract*— The *paper examines different models to automate the identification and redaction of sensitive information using Named Entity Recognition on various social media sites. We have considered a person's credit card information, bank details, home address, phone number and any sort of hate speech or offensive remark as regrettable which we aim to warn users with explanation and recommendation.

Keywords— Entity Model, BERT, Regular Expressions

INTRODUCTION

Nowadays, we have been seeing an increasing number of cases of leaks of personal information of individuals. These leaks generally occur on social media platforms such as X, Facebook, Instagram, etc. Since even a minor leak of personal information can put an individual in a compromising position, and the process of going through a corpus to match every set of sensitive information is a painstaking and arduous process, we require fast automation processes to extract said sensitive information for us. Consequently, we have also seen an increasing rise in hateful speech and offensive speeches often targeted at a particular individual or a racial or ethnic group, which often initiates unnecessary communal tensions. To counter such sensitive entities from ever leaking onto public platforms, we have aimed to redact such information through Natural Language Processing methods.

Some related works explored with respect to sensitive information redaction by [1] emphasize the necessity for robust privacy-preserving mechanisms to identify sensitive information. [2] illustrates how the anonymity factor influences user behavior and content sharing on social media platforms these days by gathering a large-scale dataset from an anonymous social media platform. [3] examines significant leaps made in NLP technologies such as BERT, which is a model that uses deep bidirectional transformers to drastically improve context understanding in language tasks. Similarly, [4] discusses GloVe, an algorithm for efficiently learning word vectors that capture precise syntactic and semantic word relationships.

I. SYSTEM ARCHITECTURE

As seen from Figure 1, our system takes both an image and a text input from the user. Both these inputs are passed through the Information extraction module where the relevant information is extracted and sent for pr-processing. After the clean text is obtained, we send it to the 2 crucial modules around which our system is built, namely Sensitive Information Detector and a classifier for Hate and Offensive texts. The outputs obtained from the sensitive information module are sent to the Sensitive information removal module and the outputs from the Hate/Offensive classifier are sent to the Recommendation engine. The outputs obtained from both the modules are then returned to the user.

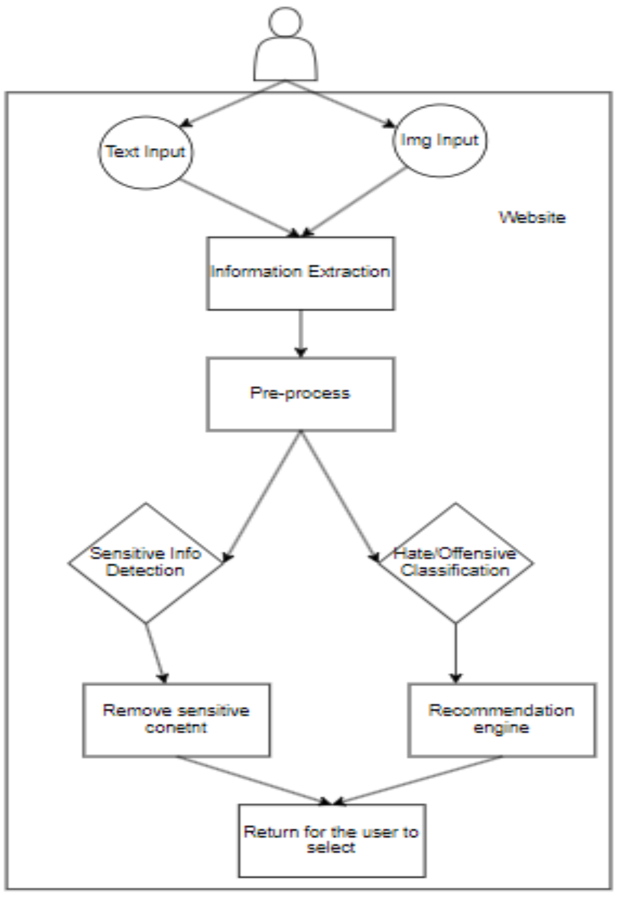


Figure 1. System Architecture

II. DATA EXTRACTION

Image: The Tesseract Optical Character Recognizer is used for the detection of words and phrases from the image input.

Confidence threshold of 60 is used. This means that only the words that the OCR is at least 60% sure are extracted and stored.
Text: The full text passed is used as data and is further cleaned along with the data extracted from image during pre-processing.

III. Exploratory Data analysis

In our project, the dataset we have used is for Hate/Offensive Speech recognition.
The distribution of class labels in the Hate/Offensive dataset can be seen in Figure 2:

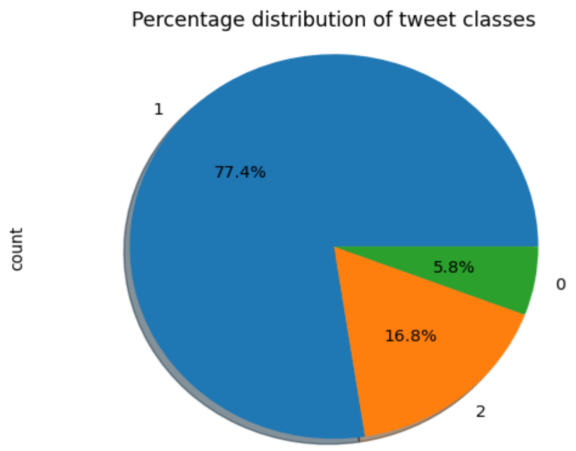


Figure 2. Class Distribution

The class labels represent:

- 0 – Hateful – 1430 samples
- 1 – Offensive - 19190 samples
- 2 – Neither – 4163 samples

We can infer from Figure 2 that class label 1 consumes ~ 80% of the full dataset. Hence training on the dataset in this state will lead to creating a bias for the model where it will overfit on label 1 and underfit on the remaining 2 labels. To tackle this problem, we use 2 solutions:

- Oversample the class samples that are in the minority for a balanced dataset
- Adjust class weights for training. Here when we are building the pre-trained models, we adjust the class weights to prevent overfitting on the class with the majority of samples.

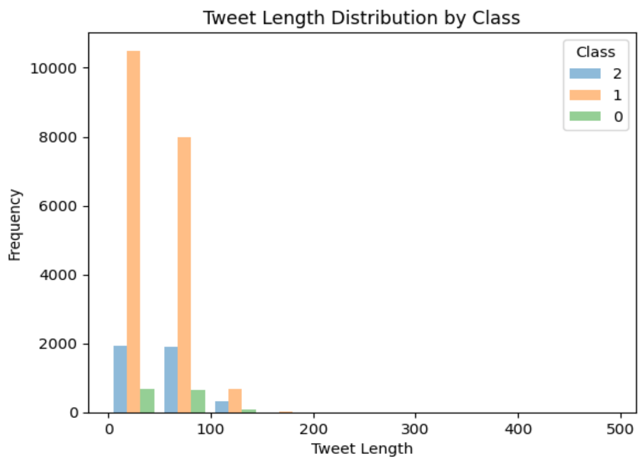


Figure 3. Tweet Length Distribution by Class

Figure 3 groups the classes based on the lengths of their associated tweets. We can see that label 1 (offensive) tweets are on average longer than the other two classes. The reason might be that people generally tend to use long texts for cursing or go huge rants when they are upset about something. This kind of pattern too is a useful indicator for learning patterns on data for generating embeddings.
Figure 4 visualizes the most frequent words in the full dataset. However, the words in this graph are mostly contributed by Class 1. Hence, we do a class distribution which can be seen in Figures 5, 6, and 7 for a better understanding.

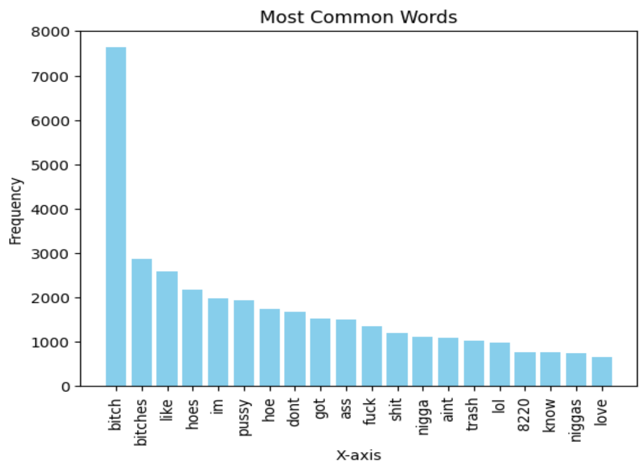


Figure 4. Most Frequent Words in Dataset

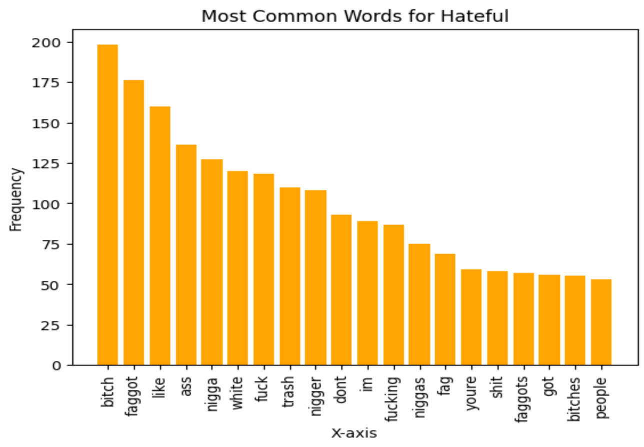


Figure 5. Most Frequent Words in Hateful Class

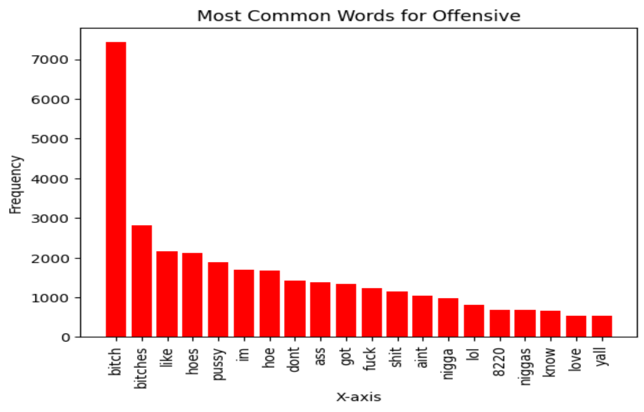


Figure 6. Most Frequent Words in Offensive Class

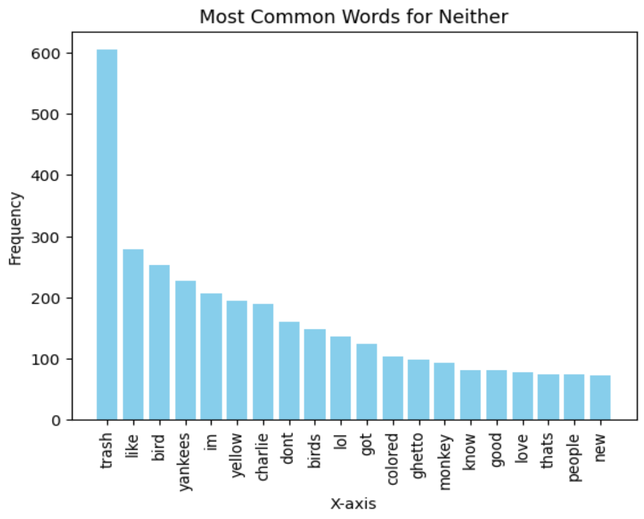


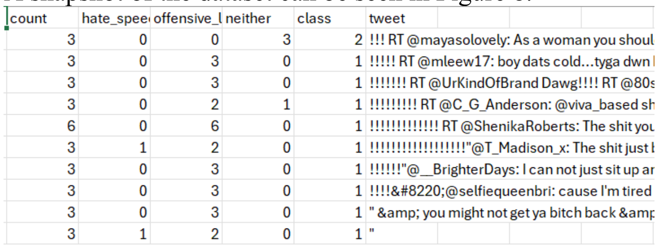
Figure 7. Most Frequent Words in Neither Class

We can infer from the graphs above that although there is some overlap between the samples in classes 0 and 1, there is still uniqueness in the results for all the classes. Where class 1 mostly consists of offensive and curse words, class 0 contains racial slurs and other hateful stuff. Class 2 on the other hand is completely different to the first two and contains no offensive and hateful words. These kinds of patterns will be learned by our models and in the embeddings. The uniqueness in features of the classes plays a pivotal role and for the models to learn.

We have taken conscious decision to not append/concatenate more datasets for having offensive/hateful data to reduce class imbalance or improve results. The reason behind this is that this dataset as seen in figure 8, has the counts in it. These counts are basically the number of users marking the tweet as hateful, offensive or neither. This allows us to analyze the results of the model. For which test sequence, does the model give bad results? Maybe the sentence had equal counts of offensive and hateful leading to the model being confused. This back analysis is important and is the reason for us not merging another dataset with this one (This is further discussed in the analysis).

IV. Pre-processing

A snapshot of the dataset can be seen in Figure 8.



count	hate_spee	offensive_l	neither	class	tweet
3	0	0	3	2	!!! RT @mayasolovely: As a woman you shoul
3	0	3	0	1	!!!!! RT @mleew17: boy dats cold...tyga dwn
3	0	3	0	1	!!!!!!! RT @UrKindOfBrand Dawg!!!! RT @80s
3	0	2	1	1	!!!!!!!!! RT @C_G_Anderson: @viva_based sh
6	0	6	0	1	!!!!!!!!!!!!! RT @ShenikaRoberts: The shit you
3	1	2	0	1	!!!!!!!!!!!!!!!!!!"@T_Madison_x: The shit just
3	0	3	0	1	!!!!!!"@__BrighterDays: I can not just sit up ar
3	0	3	0	1	!!!!&#8220;@selfiequeenbri: cause I'm tired
3	0	3	0	1	" &amp; you might not get ya bitch back &amp
3	1	2	0	1	"

Figure 8. Dataset

We can see that a variety of sentences have words like RT (retweet), @, &amp and many special characters which do not add anything to the meaning of the sentence. Hence, we remove all of these useless characters and phrases using the pipeline in Figure 9.

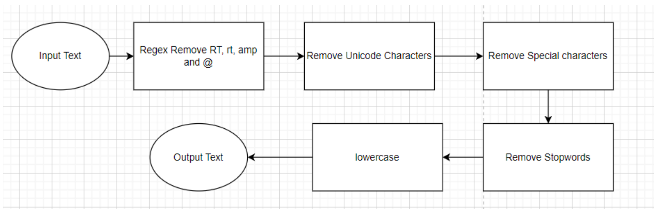

Input Text
Regex Remove RT, rt, amp and @
Remove Unicode Characters
Remove Special characters
Remove Stopwords
lowercase
Output Text

Figure 9. Pre-processing

We have regex to substitute all the useless characters as stated above which do not add anything to the semantic meaning of the sentence. After which we remove the Unicode characters by encoding the sentence with ascii characters and then decoding. Lastly, we remove the stopwords from the sentences and convert them to lowercase.

These clean texts are used by both the traditional ML algorithms and neural networks for training. However, we do not use them for training the pre-trained models. This is done as it is recommended by transformers documentation itself, to preserve the structure of the sentence.

V. Hate/Offensive speech detection

i. *Models Used*

a) *Traditional ML Models*

After cleaning the text, we started the exploration with tf-idf and count vectorizer for SVM, Naive Bayes and Random Forest models. Multiple n-gram range were tried for tf-idf and count vectorizers. The results can be seen in table 1. These results reflect tf-idf vectorizer for n-gram range of (1,2) for which the best accuracy was achieved. Even though SVM gives 89.02% validation accuracy (data was split into 80-20 for training and validation), the f1-score for label 0 (hate) is low (0.29) making the macro average of f1-score low. This is because of the low data distribution for class 0 and the models are unable to learn the concept. Although these simplistic models give decent results, they struggle massively for Out of Vocabulary (OOV) words. This is a huge concern because often on social media sites, there are many typos, and some words are elongated to bring out the exaggeration (Ex: 'lolllll!'). Also, as tf-idf vectorizer is used, the sequence of the words in the input text is lost. As we have learnt, "order matters". This is the reason we move on with testing on LSTM models where sequence/order of words is taken into account when making embeddings.

Table 1 Performance of traditional ML models

Model	Validation Accuracy	Weighted F1-score	Macro F1-score
SVM	89.02%	0.89	0.7
Naive Bayes	86.52%	0.86	0.65
Random Forest	88.33%	0.88	0.68

b) Neural Networks

We now train an LSTM (Long-Short Term Memory) [5] model and try to achieve better results than the traditional models. LSTM makes a good choice for training on time series and also text data as the order of features matters in both. This isn't considered by the traditional models where it is assumed that the features are independent of each other.
LSTM, like basic RNNs, pass the information to the neurons in the same hidden layer thus leading to sequential learning which is not possible in vanilla neural networks as they assume all input features are independent of each other. The architecture of the model we have used can be seen in Appendix Figure 1.

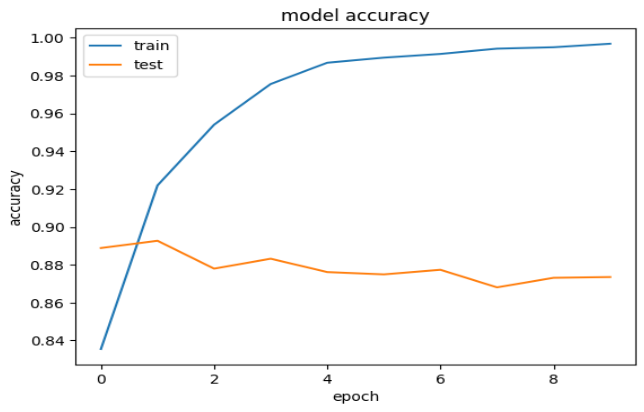


Figure 10. LSTM Training and Validation Accuracy vs Epochs

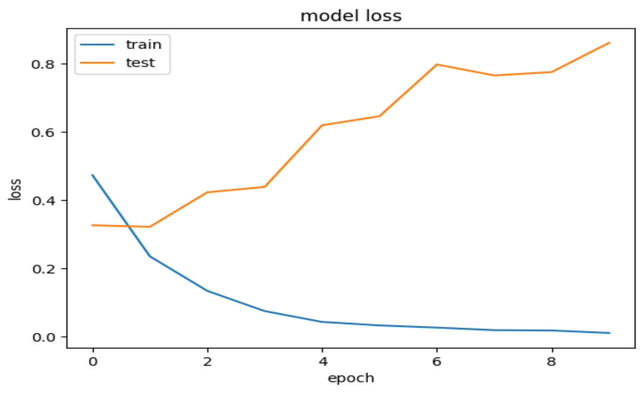


Figure 11. Training and Validation Loss vs Epochs

We can see Figures 10, that the highest model test accuracy is obtained at the 2nd epoch, after which the model starts overfitting as the validation loss increases and the training loss continuously decreases. Hence, we save the best model, and the result of the model can be seen in Table 2.

Table 2. LSTM Model Results

Training Accuracy	Validation Accuracy	F1 Score
92.19	89.27	.8938

As expected, higher scores are obtained using a neural net. However, as the Embeddings are being learned while training, the model will still fetch poor results on out-of-vocabulary words and unseen sentence structures. It is restricted to the corpus vocabulary it is trained on. Therefore, now we move on to pre-trained models where the embeddings are made on huge corpus taking sequence of words into consideration.

c) Pre-Trained Models

Comprehending underlying intentions expressed through text can uncover a vast number of valuable insights. As detailed before, the main drawback of using models and training it on the particular dataset is not being able to handle OOV. Pre-trained models are trained on large corpus of data handling OOV to a great extent. In addition to that, they are developed using state of the art architecture. They should however be fine-tuned to the use case as they are built for many NLP tasks. We will be fine-tuning these models for the text classification task. [6] gives a comprehensive methodology for achieving this task.
We will be using BERT (Bidirectional Encoder Representations from Transformers), DistilBERT, RoBERTa (A Robustly Optimized BERT), XLNet.

- BERT is a transformer-based model developed by Google in 2018. It understands the context of words from both sides (left to right and right to left) by pre-training on a large corpus of text. It consists of multiple layers of self-attention mechanisms which assign scores to tokens and feed-forward neural networks.
- DistilBERT is a distilled version of BERT, developed by Hugging Face. It is much smaller than BERT but has comparable performance.
- RoBERTa is an optimized version of BERT developed by Facebook AI. It robustly optimizes BERT by improving on pre-training process and hyperparameters to achieve better performance.
- XLNet is a generalized autoregressive pretraining method developed by Google Brain and works on permutation language modeling (PLM) objective. Here, based on all possible permutations of the input sequence order, tokens are predicted.

Training Process

PyTorch is used for training. The training process for all the models is visualized in Figure X. We will, however, be sharing the details of DistilBERT as it is the final model selected (the reasoning is explained in further sections).

- Raw dataset is used as input. No pre-processing is performed. This is because transformer-based models such as BERT utilize self-attention mechanisms to capture relationships between words in a sentence. Stopwords, punctuation, although insignificant individually, can still contribute to the context and meaning of a sentence when considered in the context of other words. Removing them may disrupt the attention mechanism and degrade model performance (punctuations can be used as emoticons).
- Then a tokenizer is used to generate encodings. 'DistilBertTokenizer' available with 'transformers' library is used (specifically 'distilbert-base-uncased'). Tokenizer splits the texts into words and then converts it to a numerical representation suitable for the use of models. In this case, input ids and attention masks are generated.

- Encodings are then split (80-20) as training and validation set.
- Then, to fine tune the model according to the data, data loaders are created for training and validation sets. Data loaders help to iterate data with batches while training.
- The selected pre-trained model is loaded with Adam optimizer by specifying 3 labels as the output. ‘DistilBertForSequenceClassification’ pre-trained model which is used for sequence classification tasks was employed (specifically 'distilbert-base-uncased').
- Hyperparameters such as learning rate and batch size are set.
- The model is trained until convergence criteria is met. If the validation loss keeps increasing for 3 times, then training is stopped. Based on the validation results, the model is re-trained with different hyperparameters.

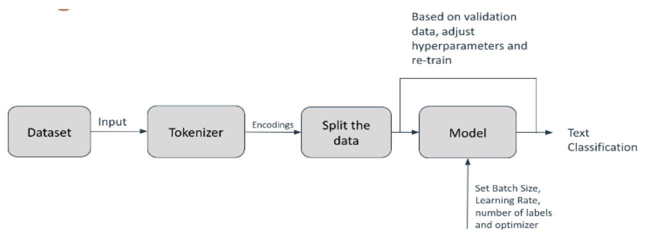

Based on validation data, adjust hyperparameters and re-train
Dataset
Input
Tokenizer
Encodings
Split the data
Model
Text Classification
Set Batch Size, Learning Rate, number of labels and optimizer

Figure 12. Training process for pre-trained models

Model Selection, Evaluation and Fine-Tuning

Table 3 presents the results for the pre-trained models. As can be seen, RoBERTa model performs the best followed by BERT and DistilBERT in both validation accuracy and f1-score. However, all the models give comparable performance. Training and loading these models take significant time. Because of which, it was decided to move with DistilBERT model as it is only about 53% of the size of RoBERTa (255 MB vs 476 MB) but gives comparable results. This saves significant training time as we explore fine tuning options (25 minutes per epoch for DistilBERT vs 44 minutes per epoch for RoBERTa with T4 GPU).

Table 3. Pre-trained model performance and size

Model	Validation Accuracy	Weighted F1-Score	Macro F1-Score	Size
BERT	91.73	0.92	0.72	418 MB
DistilBERT	91.57	0.92	0.72	255 MB
RoBERTa	91.86	0.92	0.73	476 MB
XLNet	91.38	0.91	0.71	448 MB

Figure 13 and 14 represent the training and validation losses vs epochs and training and validation accuracies vs epochs. As seen in the training graphs, the validation loss starts increasing after the 5th epoch. Co-incidentally, overfitting can also be noticed from the training and validation accuracy graph after epoch 5. Because of our convergence criteria, we retain the model after the fifth epoch. Figure 15 depicts the classification report for the validation set. As seen from the results, the f1-score for label 0 (‘hate’) is lower. This is because of the data imbalance as discussed before. One more thing to note here is that even though label 2 has less data, the model is able to learn the underlying concept as noticed from the f1-score in Figure 15. To address this, class weights of [5.0, 1.0, 1.0] were assigned, and re-trained. The validation accuracy slightly degrades (91.57% vs 90.92%) and results in lower precision for label 0 as well. However, there is a significant improvement in f1-score (0.32 to 0.49) which takes both precision and recall into consideration. The classification report for this approach can be seen in figure 16. Despite this, the results achieved are still under whelming. We also tried oversampling the data that has 0 label, but even that did not yield better results. This can be attributed to the way the oversampling works which is just random sampling with replacement.

So, we decided to further investigate the data. The data is labelled as 0 (hateful) if a greater number of users mark the tweet as hateful than offensive or neither. We noticed that for many of the hateful data, there is a count for offensive language as well. For the data with only hate markers (0 count for others), the model is able to predict such data as hateful with a much higher accuracy.

Also, since many texts can be considered both offensive and hateful, we decided to combine these for warning users, mitigating this issue to a certain extent (The user will be warned ‘your post can be considered as hateful/offensive’).

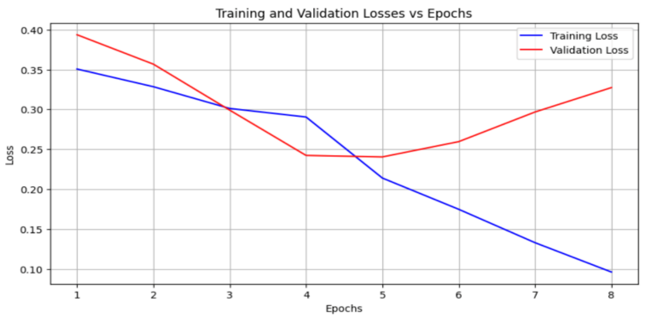


Figure 13. Training and Validation Loss vs Epochs

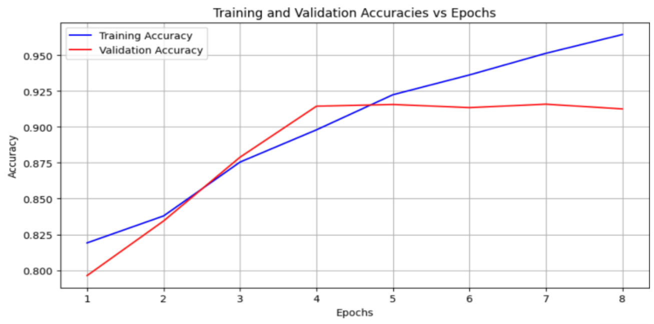
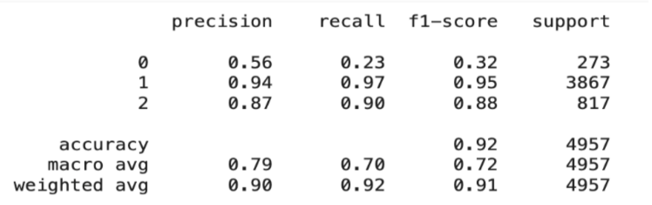


Figure 14. Training and Validation Accuracy vs Epochs

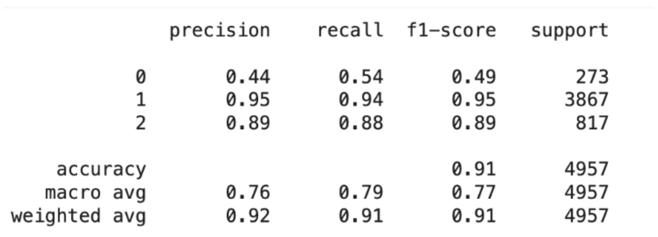


```
              precision    recall  f1-score   support

           0       0.56      0.23      0.32       273
           1       0.94      0.97      0.95      3867
           2       0.87      0.90      0.88       817

    accuracy                           0.92      4957
   macro avg       0.79      0.70      0.72      4957
weighted avg       0.90      0.92      0.91      4957
```

Figure 15. Classification report without using class weights for DistilBERT model



```
              precision    recall  f1-score   support

           0       0.44      0.54      0.49       273
           1       0.95      0.94      0.95      3867
           2       0.89      0.88      0.89       817

    accuracy                           0.91      4957
   macro avg       0.76      0.79      0.77      4957
weighted avg       0.92      0.91      0.91      4957
```

Figure 16. Classification report with using class weights for DistilBERT model

Analysis of the results

As discussed before, the primary limitation with traditional ML models is handling OOV. This is crucial for our use case as there are new words being invented every now and then (Ex: situationship for a type of relationship, zey/zem pronouns). However, depending on the embedding, the traditional models still give good results as seen in the results earlier. Even though quantitatively pre-trained models give only a slight improvement in performance, the real advantage can be experienced in real-world examples. For instance, the text '[ethnicity] people are absolute trash' can be considered as both hateful and offensive. DistilBERT model correctly predicts this text as 'offensive' whereas SVM and LSTM model classify this as 'neither'. This is because the word [ethnicity] was OOV, and the text did not contain slang words (the dataset mostly contained slang words for hateful and offensive data). Pre-trained models handle OOV better not only because they are trained on large corpus of text, but because they also understand the sentence structure which is apparent in the above example. This makes pre-trained models qualitatively much better. Also, typically very little or no pre-processing is needed, making the process faster. Pre-trained models, however, lack explainability because of the complex architecture and are very time consuming for both loading and training the model making them computationally intensive. That is why, DistilBERT model was selected as a trade-off between performance and size.

VI. Sensitive Information detection and classification

Building on our initial efforts to extract relevant text, this phase of the project focuses on the detection of sensitive information within that text. We employ a combination of Regular Expressions (Regex) and SpaCy's English NLP model to efficiently and accurately identify and categorize sensitive data. Regex is utilized for its precision in identifying structured patterns, such as contact details and financial numbers, while SpaCy's NLP capabilities allow us to detect more nuanced and contextually embedded information like personal names and locations. This approach sets the stage for the next steps in the project, which involve developing strategies for the safe handling and modification of detected sensitive information.

i. *Defining Sensitive Information*

In this project, sensitive information is defined as any data that can be used on its own or in conjunction with other data to identify, contact, or locate a single person, or to identify an individual in context. This broad category includes personal identifiers such as phone numbers, email addresses, and national identification numbers like NRIC numbers. It also encompasses financial details such as bank account numbers, PIN codes, credit card information, and security codes (CVV). This comprehensive understanding of sensitive information guides our detection methodologies and ensures a thorough safeguarding of privacy across processed text documents.

ii. *Methodologies Employed*

Our approach integrates two main techniques to enhance the detection and categorization of sensitive data:

a) *Regular Expressions (Regex)*

It's efficient in pinpointing well-defined data patterns within text. Regex is particularly adept at identifying structured information such as phone numbers, email addresses, national identification numbers (NRIC), bank account details, PIN codes, and credit card numbers including CVVs. This method provides a high degree of accuracy for these data types, which are typically formatted in predictable patterns, making them ideal candidates for Regex-based detection. To ensure robustness and reliability, these Regex patterns have been thoroughly tested across multiple inputs and edge cases, addressing common variations and formats to enhance detection accuracy.

Table 4. Regular Expressions

Data Type	Regular Expression Pattern
Phone Numbers	"\s(\+65)?[\s-]?\d{4}[\s-]?\d{4}\b"
Email Address	"\b[\w.-]+?@\w+?\.\w+?\b"
NRIC numbers	“\b[SFTG]\d{7}[A-Z]\b”
Bank Account	“\b\d{10,12}\b”
Pin codes	"\b\d{5,6}\b"
Credit Card	"\b(?:\d{4}[\s-]?){3}\d{4,7}\b"
CVVs	"\b\d{3}\b"

Table 4 includes some of the common patterns used in our Regex implementation to detect various types of sensitive information. Each pattern is tailored to match the general format of its corresponding data type, providing a foundational layer of data identification and protection.

b) *Natural Language Processing (NLP) with SpaCy*

After utilizing Regex for structured pattern detection, we integrate SpaCy's [7] English NLP model to address more complex and nuanced sensitive information within text. This method leverages Named Entity Recognition (NER) to detect and classify textual entities into several categories based on specific tags. For example, the tags `LOC` (Location), `GPE` (Geo-Political Entity), `ORG` (Organization), and `FAC` (Facility) are primarily used to identify potential addresses. However, these are just a subset of the tags employed; our methodology also encompasses other classifications to capture a broader range of sensitive data types. There are certain cases where the model fails to detect more complicated or unusual names/addresses, because the model might have never seen something similar. To alleviate this issue, we also leverage Part-Of-Speech tagging to better differentiate and categorize entities such as names and proper nouns. SpaCy's NLP model excels in its ability to contextualize and discern the semantic relationships within the text, which enhances the accuracy of detection by reducing false positives and ensuring that only relevant sensitive information is identified. This approach allows for a more comprehensive analysis and categorization of sensitive information, thereby significantly improving the project's capability to protect personal and financial data in text documents.

iii. Comparative Analysis

Regular Expressions (Regex) and SpaCy's Natural Language Processing (NLP) model were the primary methods employed in our project to detect sensitive information in text documents. Regex is highly efficient at identifying structured patterns such as phone numbers and bank details but struggles with context, leading to potential misses or false positives. In contrast, SpaCy's NLP model excels in its ability to interpret context and semantics, enhancing accuracy in detecting nuanced information like personal names and addresses. While other NER methods were also tested, SpaCy provided the most consistent and accurate results. By integrating these approaches, we capitalize on Regex's speed and precision alongside SpaCy's contextual intelligence, creating a robust system for comprehensive sensitive information detection.

iv. Limitations and Possible Solutions

While our methodologies for detecting sensitive information in text documents—Regular Expressions (Regex) and SpaCy's Natural Language Processing (NLP)—are effective, they each have limitations that could be addressed to improve overall system performance. Our approach is currently limited to Singapore specific patterns.

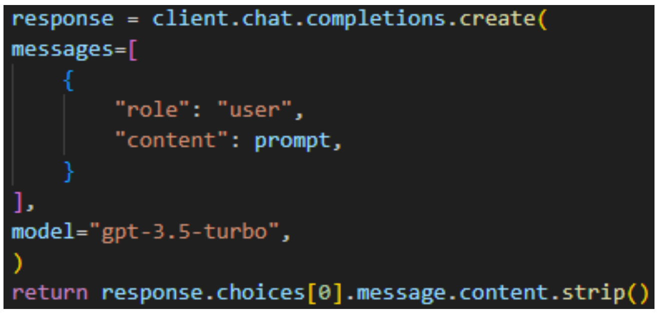
Regex excels in identifying clear, structured patterns but struggles with variations that deviate from these patterns, potentially missing critical data. To overcome this, we propose incorporating machine learning algorithms to enhance its adaptability. This would allow dynamic learning from new inputs and variations, expanding its detection capabilities without compromising speed. On the other hand, SpaCy's NLP model excels in identifying common proper nouns but may falter with unusual or rarely seen ones, resulting in missed sensitive information. This limitation arises from the model's dependency on its training data, which might not include sufficient examples of less common entities. To enhance detection accuracy, integrating a part-of-speech (POS) tagger [8] can help pre-identify potential entities for NLP analysis, ensuring even rare proper nouns are examined. Additionally, expanding and regularly updating the training dataset with a diverse range of names and terms will improve the model's ability to recognize and categorize a broader spectrum of sensitive information.

A further step to enhance reliability is to develop a verification mechanism where outputs from both Regex and SpaCy are cross validated. This hybrid approach would reduce false positives and ensure a higher accuracy rate by leveraging the strengths of both methodologies. Implementing these solutions will not only address the current limitations but also significantly improve the robustness and efficiency of our sensitive information detection system.

VII. Recommendation System

Once we have classified whether the text is offensive or not, we want to recommend a better sentence to the user without the offensive words. This is done by using **GPT 3.5** API from OpenAI. GPT is used in place of in-house models built from scratch because it is a state-of-the-art LLM (Large Language Model) built on a very large corpus. It uses a Neural Network based transformer architecture and it was very unlikely that a model built by us would generate better responses. This is the reason we went ahead with GPT.

Assuming that “txt” text is passed by the user as input and identified as offensive text. The query passed is:



```
response = client.chat.completions.create(
messages=[
    {
        "role": "user",
        "content": prompt,
    }
],
model="gpt-3.5-turbo",
)
return response.choices[0].message.content.strip()
```

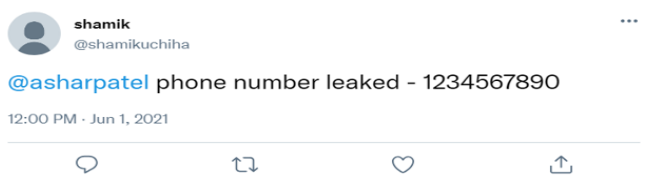
Figure 17. Query Passed to OPENAI API

Where the prompt is 'Return the sentence "{txt}" after removing the offensive words but keep the context and meaning of the sentence the same.' The recommended text is returned to the user for selection.

VIII. Sensitive content redaction

After our model has detected the sensitive content, the user is presented with the option to redact the sensitive content. The redaction process begins with loading the tweet image using Python’s PIL (PILLOW) library. The Tesseract library (Optical Character Recognition), then processes the image to return a dictionary of the sensitive words and its bounded box coordinates. We lastly load the original image for editing using OpenCV, which is a library for handling image operations. For each sensitive word in the dictionary,

It applies a Gaussian blur over the area where the word appears using the coordinates stored in the dictionary by extracting the ROI (Region of Interest). Finally, it displays the blurred image and helps to protect sensitive information from being visible in the image.



shamik
@shamikuchiha

@asharpatel phone number leaked - 1234567890

12:00 PM · Jun 1, 2021

Figure 18. Sample Input

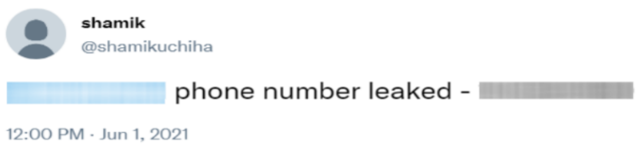

shamik
@shamikuchiha
[redacted] phone number leaked - [redacted]
12:00 PM · Jun 1, 2021

Figure 19. Output

IX. Website Architecture

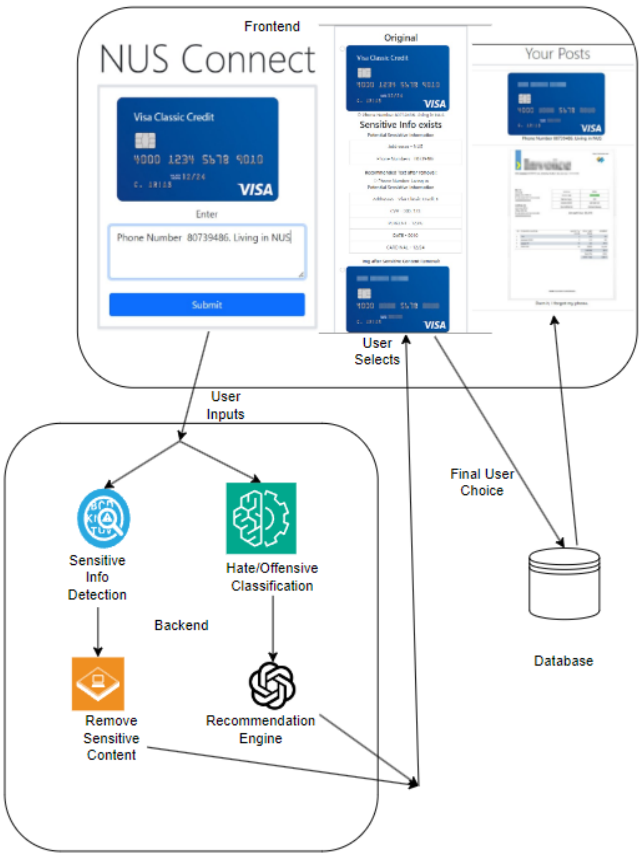


Figure 20. Flask Website Architecture

Figure 20 depicts the deployment of our Text Mining Methodologies as modules in the backend of a Flask website. As soon as the user posts an image and/or text, our modules (sensitive info detection and offensive/hate speech detection) run on the inputs and return the expected answer. The outputs are then returned to the user for selection. The user is then free to choose which of the two, original post or recommended post, does he/she want to post. The user's final post is saved in the database and can be viewed under "All Posts" tab.

X. Conclusion

This project aimed to tackle the complex problem of warning users with sensitive information and hateful or offensive speech to avoid undesired consequences. Both simplistic and complex methodologies were explored and accordingly incorporated. The project addresses what might be considered as 'regrettable' content and provides users with proper explanation along with the recommendation to alter text. This makes the user accountable for the subsequent action taken and also preserves the freedom and privacy rights of the user.

XI. Acknowledgment

We extend our sincere gratitude to our course lecturer, Christian Von Der Weth, for his continuous guidance and support. His continuous assistance and invaluable mentorship have greatly contributed to our learning experience, and we immensely appreciate his dedication and patience.

XII. References

[1] Livio Bioglio and Ruggero G. Pensa, "Analysis and classification of privacy-sensitive content in social media posts," EPJ Data Sci., vol. 11, no. 1, p. 12, 2022. PMCID: PMC8892403, PMID: 35261872.

[2] M. Mondal, D. Correa, and F. Benevenuto, "Anonymity effects: A large-scale dataset from an anonymous social media platform," in Proceedings of ACM HT 2020, virtual event, New York, USA: ACM, 2020

[3] J. Devlin, M. Chang, K. Lee, and K. Toutanova, "BERT: pre-training of deep bidirectional transformers for language understanding," in Proceedings of NAACL-HLT 2019, 2019.

[4] J. Pennington, R. Socher, and C. Manning, "Global vectors for word representation," in Proceedings of EMNLP 2014, pp. 1532–1543, 2014

[5] S. Hochreiter and J. Schmidhuber, "Long Short-term Memory," *Neural Computation*, vol. 9, no. 8, pp. 1735-1740, Dec. 1997. doi: 10.1162/neco.1997.9.8.1735.

[6] Sun, C., Qiu, X., Xu, Y., & Huang, X. (2019). How to Fine-Tune BERT for Text Classification. In Proceedings of the 26th International Conference on World Wide Web Companion (pp. 1371–1379). Association for Computing Machinery.

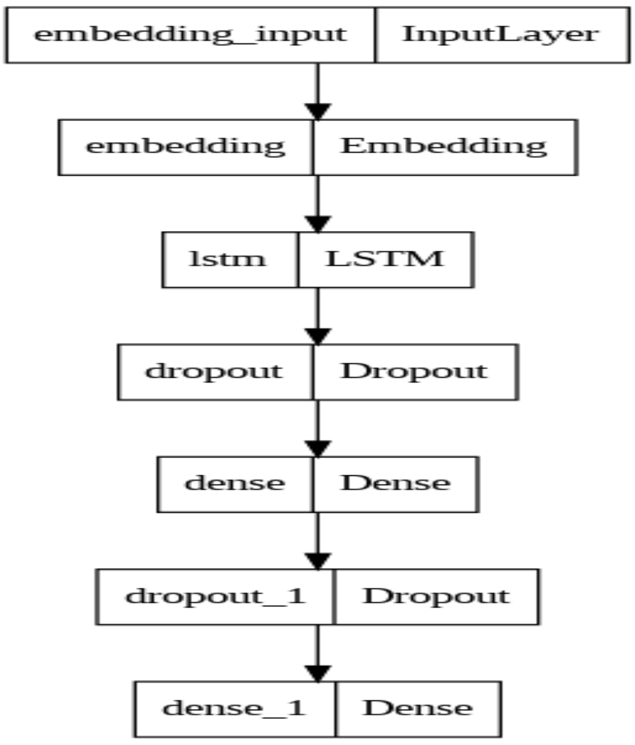
[7] M. Honnibal and I. Montani, "Advancements in NLP: Development and Application of spaCy's English Model," in Journal of Computational Linguistics, vol. 37, no. 4, pp. 841-856, Oct. 2021

[8] A. B. Smith and C. D. Johnson, "Enhancing Language Processing with a New POS Tagger," in Proceedings of the International Conference on Language Technologies, vol. 29, no. 1, pp. 202-210, May 2022

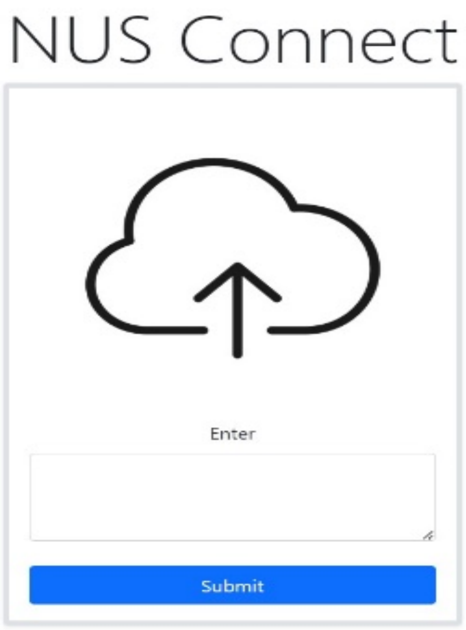
Appendix

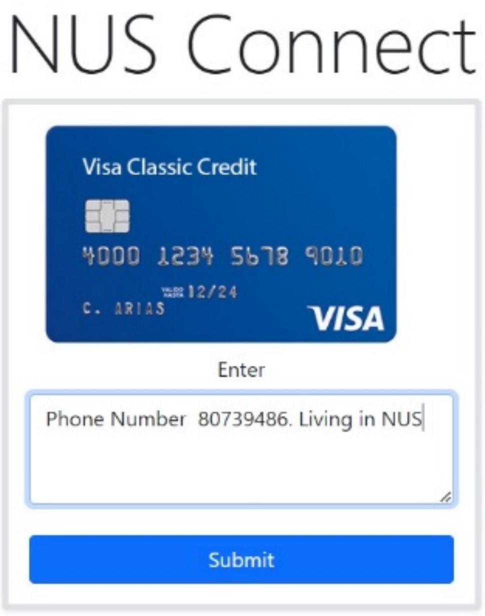
Please find our code on the GitHub repository:
https://github.com/Jas-077/TextMining.git

Figures

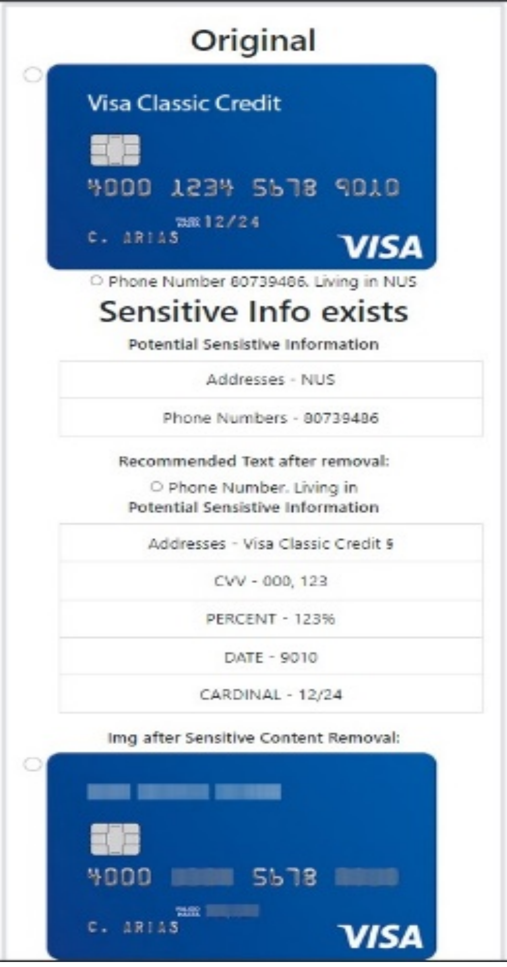


Appendix Label 1. LSTM Model Architecture





Phone Number 80739486. Living in NUS



Phone Number 80739486. Living in NUS

Sensitive Info exists

Potential Sensistive Information

Addresses - NUS
Phone Numbers - 80739486

Recommended Text after removal:

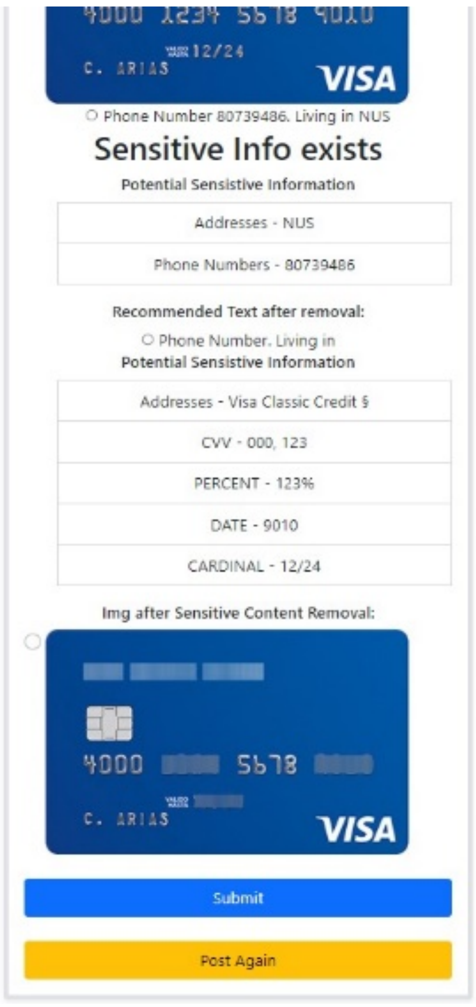
Phone Number. Living in

Potential Sensistive Information

Addresses - Visa Classic Credit 5
CVV - 000, 123
PERCENT - 123%
DATE - 9010
CARDINAL - 12/24

Img after Sensitive Content Removal:

Choose to keep the number even if detected as sensitive info.



○ Phone Number 80739486. Living in NUS

Sensitive Info exists

Potential Sensistive Information

Addresses - NUS
Phone Numbers - 80739486

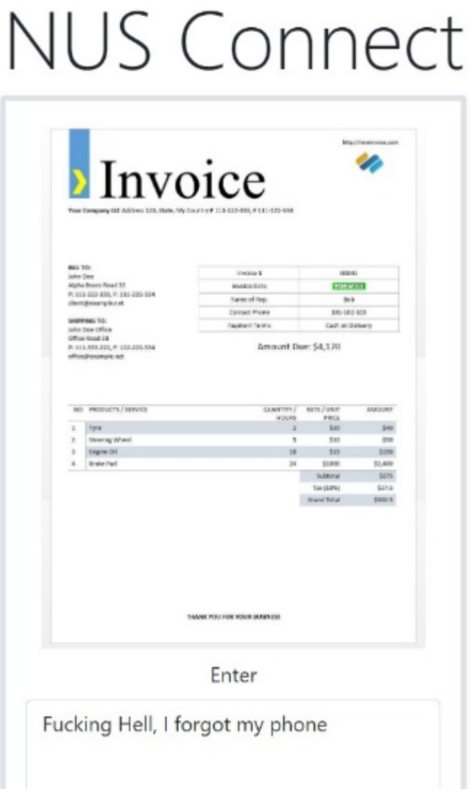
Recommended Text after removal:

○ Phone Number. Living in

Potential Sensistive Information

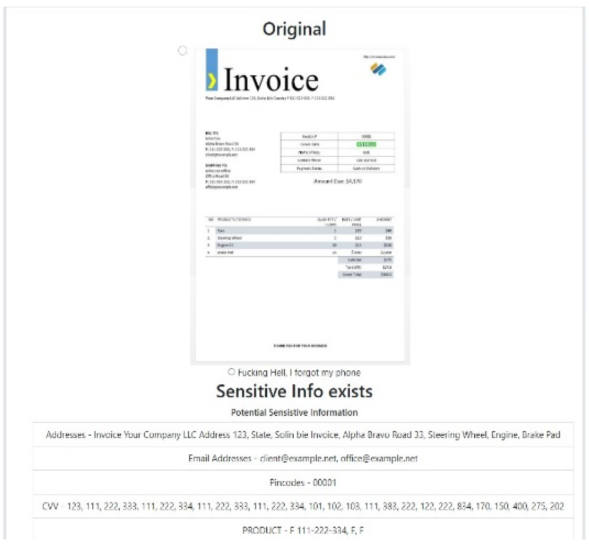
Addresses - Visa Classic Credit $
CVV - 000, 123
PERCENT - 123%
DATE - 9010
CARDINAL - 12/24

Img after Sensitive Content Removal:



Enter

Fucking Hell, I forgot my phone

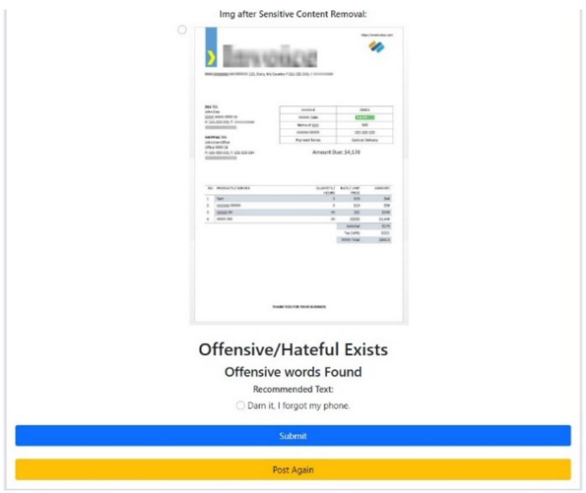


○ Fucking Hell, I forgot my phone

Sensitive Info exists

Potential Sensistive Information

Addresses - Invoice Your Company LLC Address 123, State, Solin bie Invoice, Alpha Bravo Road 33, Steering Wheel, Engine, Brake Pad
Email Addresses - client@example.net, office@example.net
Pincodes - 00001
CVV - 123, 111, 222, 333, 111, 222, 334, 111, 222, 333, 111, 222, 334, 101, 102, 103, 111, 383, 222, 122, 222, 834, 170, 150, 400, 275, 202
PRODUCT - F 111-222-334, F, F


Img after Sensitive Content Removal:

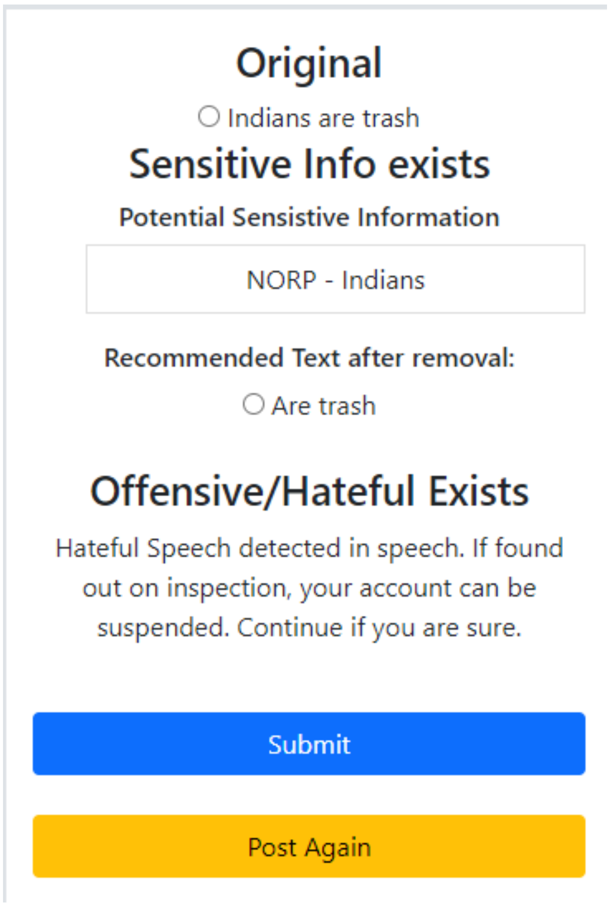
Offensive/Hateful Exists

Offensive words Found

Recommended Text:

○ Darn it, I forgot my phone.

Offensive word was removed above and a new sentence was recommended.



Original

○ Indians are trash

Sensitive Info exists

Potential Sensistive Information

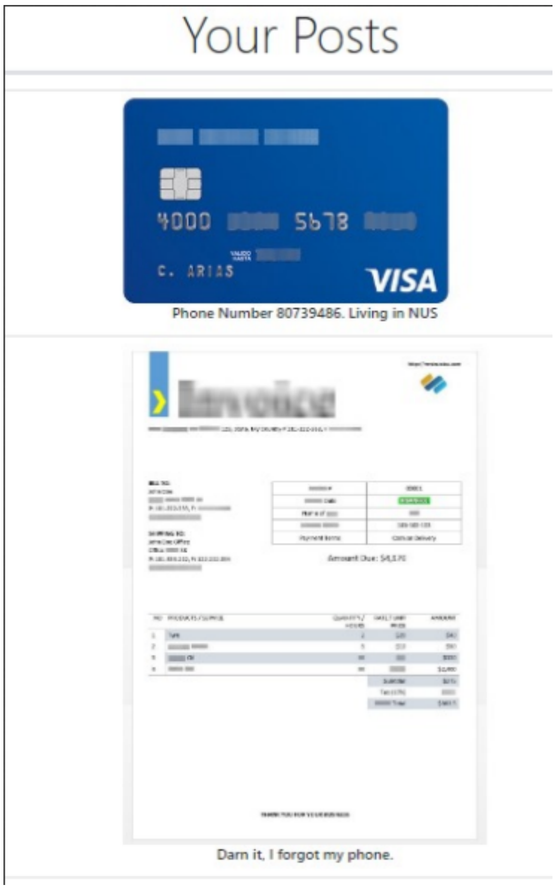
NORP - Indians

Recommended Text after removal:

○ Are trash

Offensive/Hateful Exists

Hateful Speech detected in speech. If found out on inspection, your account can be suspended. Continue if you are sure.



Your Posts

Phone Number 80739486. Living in NUS

Darn it, I forgot my phone.