2022

ECE 1786 Final Report

NOffence:
Hate Speech and Offensive Language Detection on Social Media

YIFAN CHEN 1002935817

CHUTIAN TAO 1007689431

Word Count = 1997/2000

Penalty = 0%

Table of Contents

1.0 Introduction.....1
2.0 Related Work.....1
3.0 Project Workflow.....2
4.0 Data Processing 3
5.0 Baseline Models 6
6.0 Architectures of models.....6
7.0 Results.....8
8.0 Discussions and Learnings.....9
9.0 Individual Contributions 11
References 12
Appendix.....13

1.0 Introduction

Hate Speech has drawn considerable attention in recent years due to its rapid dissemination on online social media platforms, as well as interactivities on streaming platforms and In-game chats. It is pervasive on social media and an increasingly worrying issue in society.

Per the Criminal Code of Canada, hate speeches are seen as criminal offences. For an automatic hate-speech detection program, a key challenge is to separate hate speech from offensive languages.

The goal of this project is to develop a program that recognizes hateful, offensive, or neutral speech on Twitter and other social media. Unlike conventional censorship by humans, our technique would be more responsive, more flexible, and protect users' privacy better.

2.0 Related Work

To confront toxic content, a great deal of work has been done. The upfront information to be extracted is the lexical features where the summation of word meanings is used as the meaning of a sentence [1], e.g., a bag-of-words (BoW) feature. The primitive BoW is an intuitive methodology of vectorization at cost of low computation.

In demand for more information, revisions including N-Gram and Term Frequency–Inverse Document Frequency (TF-IDF) have improved BoW [2]. N-Gram takes a sequence of N words into vocabulary instead of single words used by BoW. TF-IDF normalizes word frequencies in sentences by the word frequencies appeared in whole corpus which weakened the impact of common words.

With BoW features, machine learning algorithms such as Support Vector Machines (SVM), Random Forest, Logistic Regression, and Naive Bayes are qualified candidates to classify contents [1][2]. Burnap and Williams have discovered linear SVM output satisfying predictions [1]. Davidson, Warmsley, Macy, and Weber have compared a few models with tuned parameters (N-Gram ranges, regularizations, etc.) and came with SVM with liblinear as the best solver [2]. Davidson et al's best model has achieved an overall F1 score of 0.90, however, with significant misclassification between hate speeches and offensive languages.

Although BoW was proved useful and widely used, it has obvious limitations: changes to vocabulary will cause changes to the model structure; and it ignores the relationship between words. Therefore, NLP researchers came up with Word2Vec (Word Embeddings) technique. Rohan, Tyrus, Kathleen, and Susan performed hate speech detection using such vectorization [3]. In Rohan et al's work, their Transformed Word Embedding Model (TWEM) outperformed Logistic Regression with N-Gram and reached an F1 score of 0.924 in the Hate Speech and Offensive Language Dataset [4].

3.0 Project Workflow

The following figure shows the overall workflow for this project. We implemented three baseline models and five advanced models.

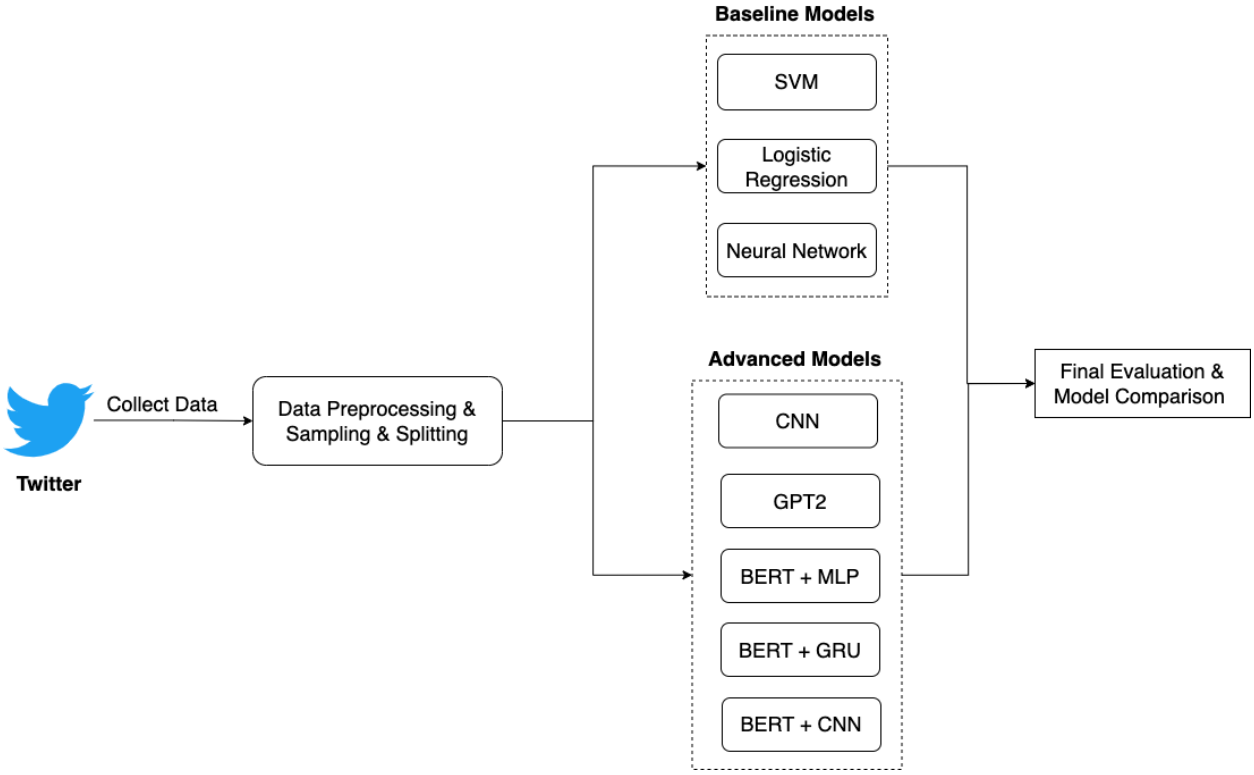

Figure 1: Overall Workflow for Project

4.0 Data Processing

To launch our project, we have started with the Hate Speech and Offensive Language Dataset [4] (dataset 1).

- Data Cleaning

  Tweets often contain more than plain text and punctuation. Thus, we prepared data for the vectorizer by removing phrases that cannot be tokenized while keeping precise meanings as much as possible. Processing steps in practice are listed below.

  - Expand abbreviations
  - Lower casing
  - Remove:
    - Stop words
    - Tweet specific phrases (mentioned “@user”, topic (#)/retweet (!!! RT) …)
    - Multiple white spaces/ break-lines
    - URLs/ “#Ampersand” reference
    - Non-ASCII characters
    - Non-alphanumeric characters
    - Other Unicode strings (Emojis…)
  - Delete very short words (length less than 2)

  Here provides an example of cleaned data and its original form:

  - Original:
    “!!! RT @mayasolovely: As a woman you shouldn't complain about cleaning up your house. &Amp; as a man you should always take the trash out...”
  - Processed:
    “woman you should not complain about cleaning your house man you should always take the trash out”

- Data Balancing and Splitting (Pre-Supplement)

  Since classes for dataset 1 are extremely imbalanced with 77.4% offensive, 16.8% neutral, and 5.8% hate, we initially balanced the classes by under-sampling the majorities, i.e., offensive and neutral speech, to prevent our model from skewing the data in favor of one of the classes.

  Then, we split data into training, validation, and test set with a ratio of 0.64:016:0.2. We have shuffled the whole dataset as well as maintained the stratification between classes.

- Data Supplement

  Shortly after implementing a few models, we found it necessary to acquire more data.

  The team experimentally hand-labeled about a few hundred tweets scrapped from the latest posts. Our experiment proved hand-labelling infeasible that it is terribly time-consuming, and we can hardly guarantee label accordance.

  Thus, we introduced two additional datasets: the multimodal hate speech dataset [5] (dataset 2) and the sentiment 140 dataset [6] (dataset 3), In such a case, we selected those tweets that are confidently identified hate speeches from dataset 2 and a small fraction of positive speeches from datasets 3.

  As a result, the smallest class, class 0 now has 10,002 records. In terms of convenience and balance, we sampled 10,000 rows from each class.

  Same as before, we identically split the new data.

- Vectorization

  With processed training, validation, and test datasets, we implanted vectorizations before fed into different models as required (TF-IDF or Word2Vec).

The following table summarized class distribution at any phase we have modified data volume.

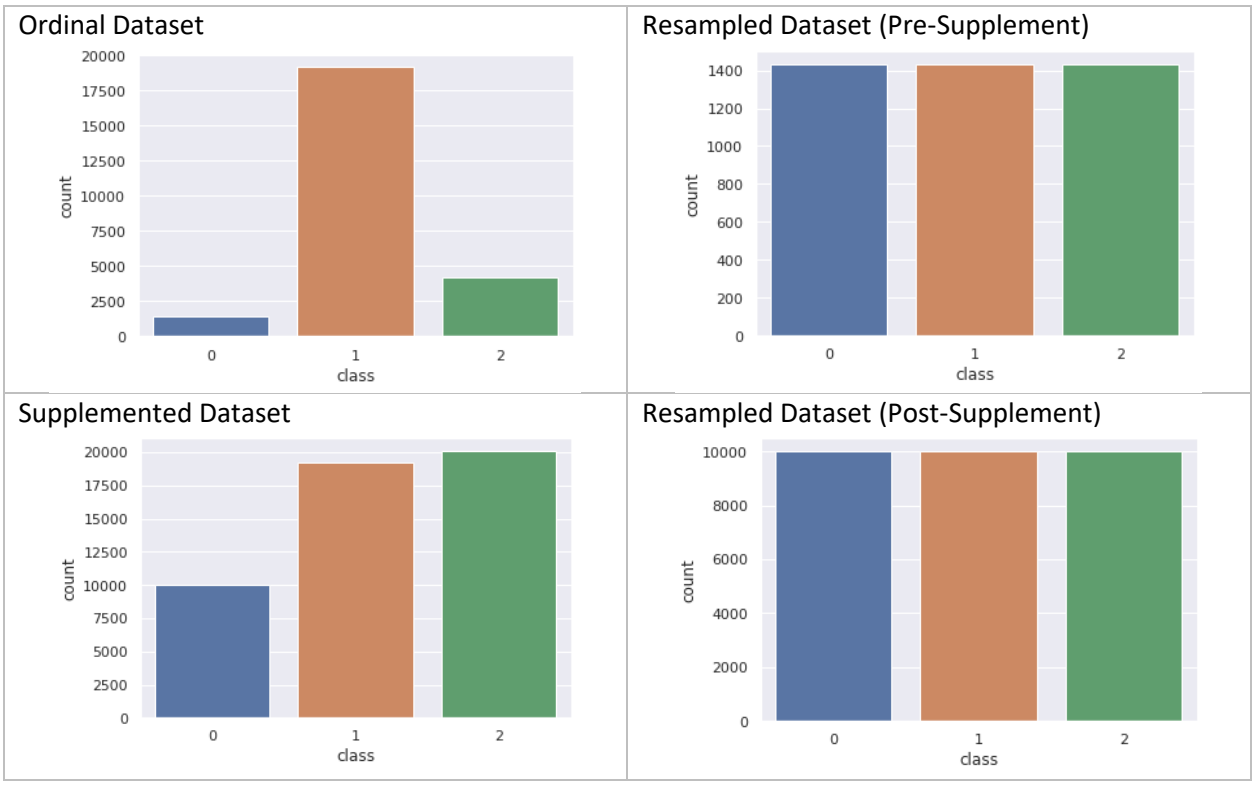

Table 1: Classes Distributions of Datasets during Processing

5.0 Baseline Models

With the processed dataset, we have implemented three baseline models.

According to the work discussed in [1] and [2], we found it effective using TF-IDF vectorizer. In practice, we limited the vocabulary size to 5,000. We have tested several models utilized in the documents [1] and [2] including k-NN, Naive Bayes, Random Forest, Logistic Regression, and SVM. Based on convergence times and classification performances, we have kept Logistic Regression and SVM for further comparison.

In addition, we have produced another baseline with Word Embeddings. We constructed a Multi-Layer Perceptron (MLP) with single embedding input, i.e., the average word embeddings within a tweet, as well as three output neurons followed by a SoftMax layer so that represent the three classes. In practice, we used pre-trained GloVe vocabulary with specified name="6B" and dim=100.

6.0 Architectures of models

In addition to baselines, we have attempted to construct several neural networks:

- CNN

  The CNN is similar to the MLP we built as a baseline; despite we replaced the “averaging” layer with a convolutional layer. We have only utilized kernels of two sizes (3 and 5) and set both output channels to 45 through rough a grid search.

- GPT2

  The GPT2 is a uni-directional transformer architecture pre-trained on a very large corpus in English. Its attention mechanisms allow the model to specifically focus on the most relevant content [7]. In practice, we load the smallest version of GPT-2 with 124M parameters from hugging face’s Transformer package: “GPT2ForSequenceClassification”, as well as the respective GPT2Tokenizer.

- BERT

  BERT is also a transformer architecture trained to learn language representations and conceived to be used as the main architecture for NLP tasks in recent years. The following picture shows the architecture of our model.

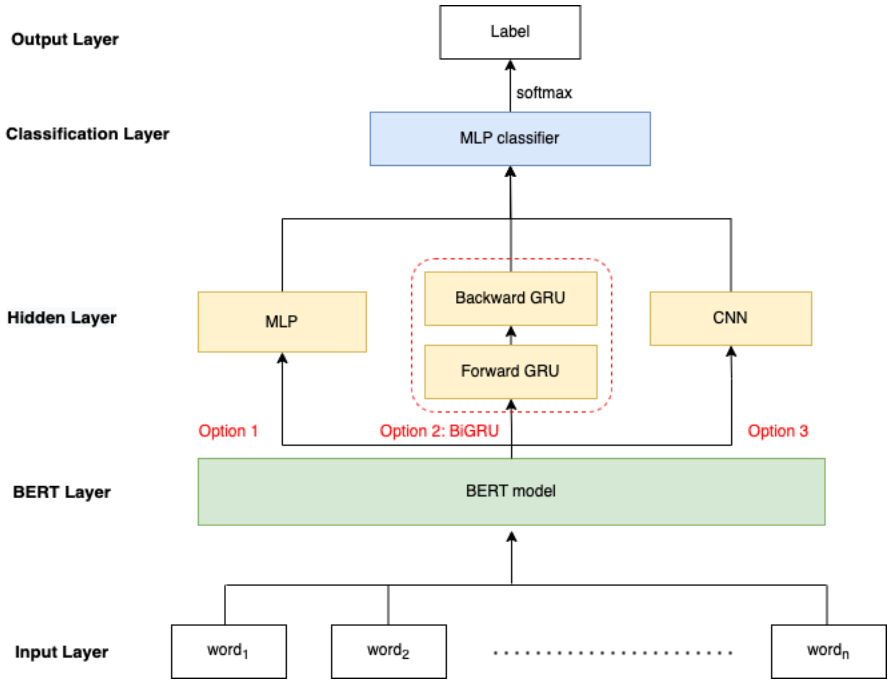

Figure 2: BERT Architectures with Potential Classification Layer

There are three hierarchies of BERT. First, the vectorized text data is fed into a pre-trained BERT layer to obtain the semantic information. After that, a hidden layer is defined to further extract the context information from the data. We designed 3 options for this hidden layer.

- The first option is only a simple layer that takes inputs of averaging word embeddings (input dimension=768, #parameters = 2,307).
- The second option we chose is a Bidirectional GRU extension (layers=2, hidden dimension=512, dropout=0.5, #parameters = 8,666,115), as it is capable of simultaneously extracting the text information features from both directions.
- The third option is a CNN hidden layer (#filters = 100, filter window sizes = [2, 3, 4, 5], dropout=0.5, #parameters = 1,076,803). It can capture spatial relationships via a sliding window.

Finally, an MLP classifier and SoftMax function are applied to generate classification labels.

7.0 Results

With training and validation sets to implement models, we have summarized the performances on the test set, and compared some models with the pre-supplement data.

Models	Vectorization	F1-Score (Class 0)	F1-Score (Class 1)	F1-Score (Class 2)	Accuracy
Baseline					
SVM (Old Data)	BoW (TF-IDF)	0.73	0.78	0.84	79%
SVM	BoW (TF-IDF)	0.90	0.88	0.95	91%
LR (Old Data)	BoW (TF-IDF)	0.73	0.80	0.87	80%
LR	BoW (TF-IDF)	0.90	0.89	0.95	91%
MLP (Old Data)	Word Embeddings	0.65	0.73	0.77	72%
MLP	Word Embeddings	0.71	0.74	0.82	78%
Main Model					
CNN	Word Embeddings	0.88	0.88	0.91	89%
GPT2	Word Embeddings	0.82	0.81	0.86	83%
BERT+MLP	Word Embeddings	0.76	0.75	0.81	78%
BERT+GRU (Old Data)	Word Embeddings	0.68	0.73	0.77	73%
BERT+GRU	Word Embeddings	0.92	0.90	0.95	93%
BERT+CNN	Word Embeddings	**0.92**	**0.91**	**0.95**	**93%**

Table 2: Test Results Summary and Comparison

As shown in Table 2, we have found that the classical approaches are competent to this task to some extent. Both SVM and Logistic Regression methods perform similarly, giving accuracies of around 91%.

A noticeable limitation to our MLP baseline is that it only takes one embedding per text to fit so it acted as a "bottleneck" (performed no better than 80% accuracy) and the convolutional layer showed significant progress than the MLP. Likewise, we observed such progress between BERT+MLP and BERT+CNN.

Table 3 shows qualitative results of three classes with predictions from BERT+CNN.

Text	Prediction	True label
trump has gone full ret*rd	0	0
even they know you b*tch	1	1
happy monday morning	2	2
** 0 – Hate speech; 1 – Offensive language; 2 – Neither/Neutral*		

Table 3: Representative Samples with True Labels and Predications

8.0 Discussions and Learnings

Overall, our best model (BERT+CNN) performs well as it has the highest accuracy while maintaining high f1 scores. The following figures show training curves of loss and accuracy for both the training and validation dataset respectively. The model keeps learning features as the number of epochs increases, with no overfitting happening.

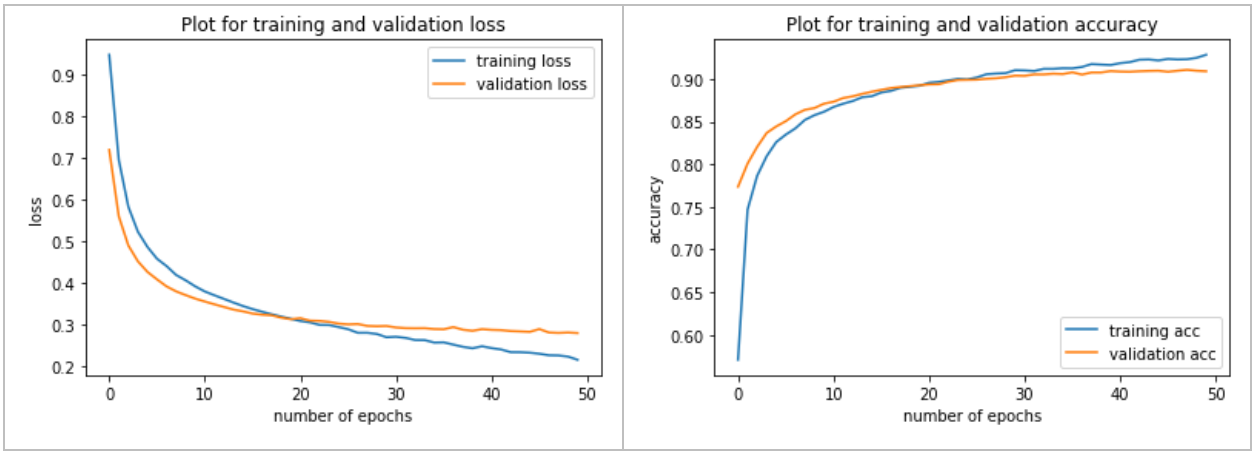

Table 4: Test Results Summary and Comparison

An impressive discovery from the results is that the machine learning baseline model also archives high test accuracy. Take the SVM for instance, it has 91% accuracy, which is close to our best model 93%, and even overperforms most of our neural networks.

According to the confusion matrix below, we can see that the BERT+CNN architecture is slightly better.

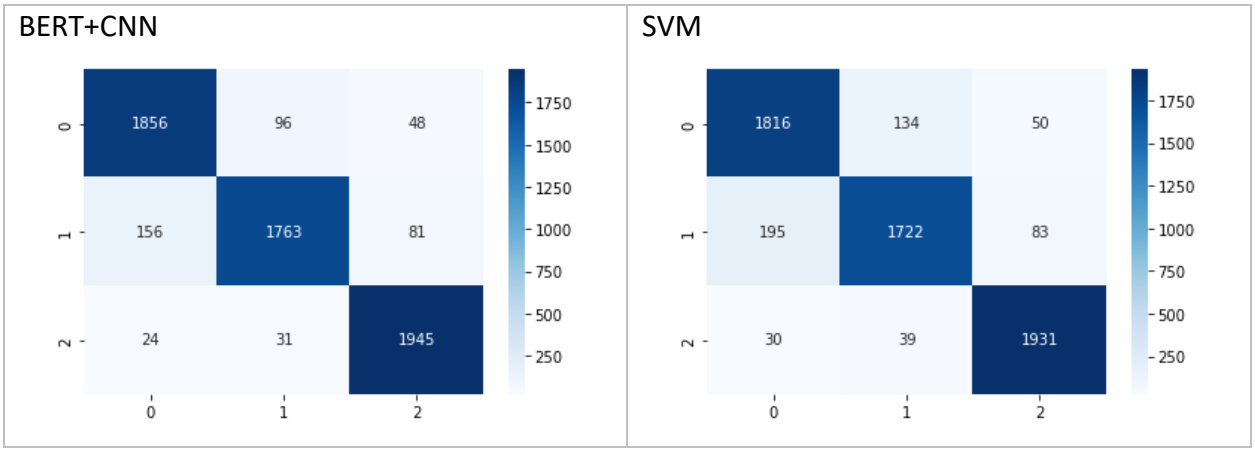

Table 5: Test Results Summary and Comparison

Another finding from the confusion matrices is that we knew the baselines are prone to mixing up hate speeches and offensive language, so we wanted to investigate if transformers can do better. As a result, transformers did reduce such mispredictions but only to some extent.

We suspected three possible reasons for such an outcome. The first might be the limited data volume, we speculate that transformers need massive data to be finetuned. The second is tweets contents have a significant number of out-of-vocabulary tokens when using pre-trained vocabulary, which cause information loss. The last reason we come up with is that those baseline models like SVM and Logistic Regression have stronger initial assumptions about the data (i.e., Inductive Bias). In comparison, Deep Learning architectures are more data-driven, which could explain classical techniques' comparable results to our transformers in this task.

For potential improvements, we would propose some measures that we would do differently in a similar project based on our experience: The first is to apply for Twitter developer account with sufficient privilege at earlier stages to access a larger amount of data. In addition, it will be beneficial to train our own vocabulary once we have enough computation power.

9.0 Individual Contributions

Over the past few weeks, the team has efficiently completed equivalent work and the collaboration between members are efficient, with meetings frequently to track the progress of the project.

Specifically, Chutian and Yifan shared data collection and processing work; and evaluated and compared models together. In addition,

- Chutian:
    - Supplemented dataset with new data
    - Implemented baseline models and GPT2 classifier
    - Assisted Yifan to debug and tune BERT models
- Yifan:
    - Balanced data with resampling
    - Implemented BERT models with classification heads (MLP, GRU, and CNN)
    - Assisted Chutian in constructing baselines and Transformer

References

[1] P. Burnap and M. Williams, “Us and them: identifying cyber hate on Twitter across multiple protected characteristics,” *EPJ Data Science*, vol. 5, Dec. 2016, doi: 10.1140/epjds/s13688-016-0072-6.

[2] T. Davidson, D. Warmsley, M. Macy, and I. Weber, *Automated Hate Speech Detection and the Problem of Offensive Language*. arXiv, 2017. doi: 10.48550/ARXIV.1703.04009.

[3] R. Kshirsagar, T. Cukuvac, K. McKeown, and S. McGregor, “Predictive Embeddings for Hate Speech Detection on Twitter,” in *Proceedings of the 2nd Workshop on Abusive Language Online (ALW2)*, Oct. 2018, pp. 26–32. doi: 10.18653/v1/W18-5104.

[4] A. Samoshyn, “Hate speech and offensive language dataset,” Kaggle, 17-Jun-2020. [Online]. Available: https://www.kaggle.com/datasets/mrmorj/hate-speech-and-offensive-language-dataset. [Accessed: 26-Oct-2022].

[5] R. Gomez, J. Gibert, L. Gomez, and D. Karatzas, *Exploring Hate Speech Detection in Multimodal Publications*. arXiv, 2019. doi: 10.48550/ARXIV.1910.03814.

[6] M. KazAnova, “Sentiment140 dataset with 1.6 million tweets,” Kaggle, 13-Sep-2017. [Online]. Available: https://www.kaggle.com/datasets/kazanova/sentiment140. [Accessed: 27-Oct-2022].

[7] A. Radford et al., “Language models are unsupervised multitask learners,” *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[8] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. arXiv, 2018. doi: 10.48550/ARXIV.1810.04805.

Appendix

Permission to post the following three items on a course website	
Chutian	
Permission to post video	wait till see video
Permission to post final report	yes
Permission to post source code	yes
Yifan	
Permission to post video	yes
Permission to post final report	yes
Permission to post source code	yes