Head off Toxic Comment in Social Media – Group 18

Authors:
Cao Huimin (A0186031M), **Yang Qing** (A0152277A), **Yu Zongdong** (A0140018X), **Zhang Meng** (A0119389N)

1. Introduction

1.1 Problem Background and Definition

The increasing popularity of social media has provided platforms for people to express their opinions. Toxic comments have become more common and so raised public concerns. A Pew survey (Rainie, Anderson, & Albright, 2017) showed that 39% of the experts expect the online future will be "more shaped" by harassment, trolling, distrust. To consider an even worse case, toxic comments may result in cyber violence. So it is important for business owners to spot those toxic comments. Many social media and live-video streaming platforms may face the difficulty of identifying toxic comments timely among a huge number of comments to be processed every moment. In this case, implementing machine learning models to predict the probability of toxicity, and generate warnings for toxic comments will help enterprises to save costs and increase efficiency as well as user experience. The main objective of our project is to use machine learning and NLP techniques to build a prediction model to detect toxic comments. Finally stop those comments from posting into the social media platform.

1.2 Our solutions

In this project, we deep dived into understanding our datasets to come up with appropriate features, and explored many classic machine learning and deep learning models. Logistic regression performs the best among classic models. For deep learning models, RNN and BERT give us good results in terms of AUC-ROC score. We will use two potential business cases as examples to explain how our solution will help the business owners. We also prepared a simple dashboard to demonstrate to them how our solution works. A Github link can be found at the end of our report.

2 Data Source and Description

2.1 Kaggle Toxic data[1]set

The Toxic Comment Dataset is a multi-label dataset. It is provided by a kaggle competition (https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge) and serves as the main dataset for training and validation of our models. The comments in this dataset are collected from the Wikipedia Talk page, where editors can share and discuss their views and improvement on Wikipedia pages and articles. Given the nature of Wikipedia, the topics of the comments can cover many sectors and disciplines. Six labels were annotated by human intervention, including toxic, severe_toxic, obscene, threat, insult and identity_hate. A total of 159,571 observations are included. There are eight variables in the dataset - the six categories of toxicity in boolean format, plus id and comment_text in string type. (Refer to Appendix Fig 1 for word cloud)

2.2 Scraped twitter dataset

Our second dataset is collected by scraping historical tweets from 01 April 2020 to 10 April 2020 that were discussing the topic of 'coronavirus', with GetOldTweets3 python package. There are a total of 100,000 observations in the dataset, averaging 10,000 tweets per day. Each observation consists of 4 features: id, date, username, text. (Refer to Appendix Fig 2 for word cloud)

3. Exploratory Data Analysis (EDA)

3.1 Exploration on Toxicity Labels

We firstly looked at how each comment is labelled and how many of records are included in every category. Our dataset is imbalanced with a much higher number of non-toxic comments than others, see *Fig 1*. Around 6% of the observations are labelled with more than one tag. Among

Master of Science in Business Analytics (MSBA)
National University of Singapore

the 16,225 derogatory comments, around 61% of them have more than one label. Intuitively, for example, if a comment is labelled as severe toxic, it should be considered as toxic as well. To have a better understanding of the label method, we then tested our hypothesis and observed that 100% of severe toxic, 93.8% of obscene, 93.9% of threat, 93.2% of insult and 92.7% of identity hate are classified as toxic at the same time.

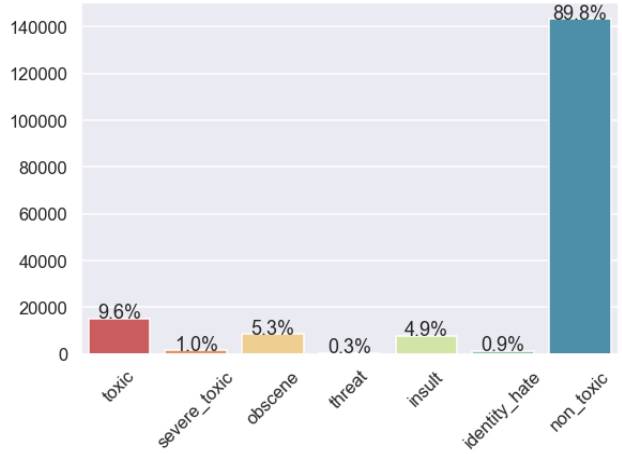

Fig 1. The percentage represents the number of labels for that category over the total number of records.

3.1.2 Exploration on Comments Properties

Considering what people will type out when they get angry, they may tend to use more capital letters, exclamation marks, question marks to express their emotion. Asterisk may also be a symbol we could have a look because it can be used as a mask of toxic words, such as sh*t. So, to test our assumptions, we calculated the percentage of capital letters, count of exclamation marks, question marks as well as asterisk in each comment. It is interesting to see that indeed toxic comments have a higher proportion of capitals, see *Fig 2*.

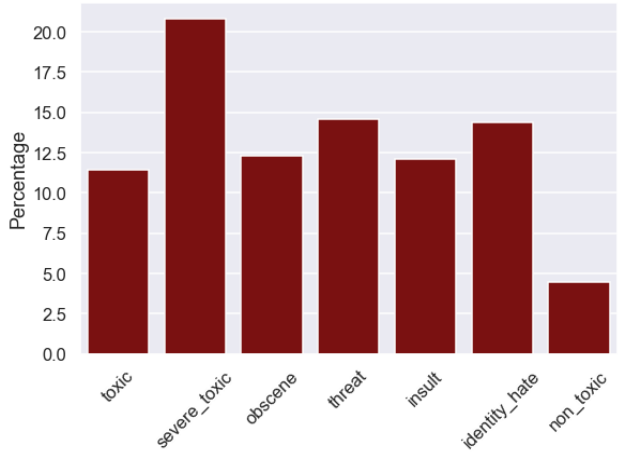

Fig 2 The average percentage of capital letters among all the characters in comment for each category.

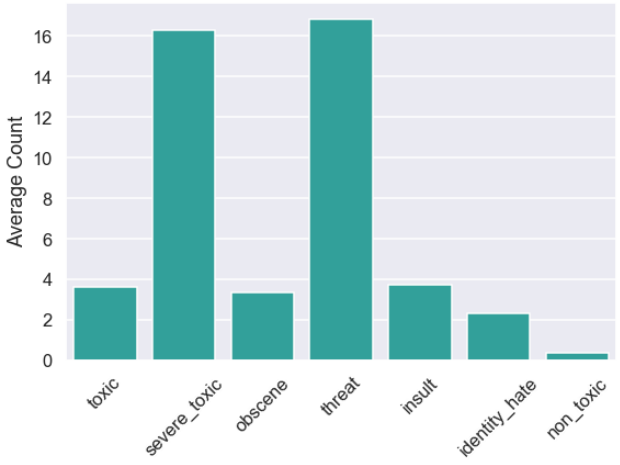

Fig 3. The average count of exclamation marks in comment for each category.

For the three special symbols, exclamation marks have high counts in severe_toxic and insult, while asterisk has a similar pattern as capital letters. The number of question marks are fairly close among different groups. From the results we can prove our assumptions that capitals and exclamation marks are being more frequently used in toxic comments.

We next removed non-letters and converted all characters into lower case. We had a few initial assumptions in mind to test, like angry people may prefer to use shorter sentences and lower variety of vocabulary. So we studied the sentiment polarity score, text length, word count, unique word count and percentage of unique words for each comment, and then took the average for each category. The sentiment score shows that non-toxic comments are in the positive territory, while the other six groups of toxic categories are all averaged to negative numbers. An interesting aspect to take note is that the severe_toxic group has the highest text length and word count, but the lowest unique words. This implies when people give extreme toxic messages, they tend to write long sentences with repetitive words. Other results, in general, indicate toxic comments usually have lower word count and unique words, as well as smaller text length. The percentage of unique words are around 80% for all groups.

Lastly, we explored the high frequently-used words in each group. We computed the word cloud before removing the stopwords. As expected, the most common words are those stopwords such as “the”, “a”, “it”, “that” etc. So it is necessary to take them out. Then we removed those stopwords and repeated the same step. Those toxic vocabularies indeed appear as the most common words in all the six groups, whereas words like “Wikipedia”, “article”, “page”, “think”, “talk” occur the most in clean

comments. We will discuss and explore more on this under our LDA analysis. Please see *Fig 4* and *Fig 5* for the world cloud of toxic and non_toxic groups.

Fig 4. Most common words for the toxic group

Fig 5. Most common words for the non_toxic group

3.1.3 Text Complexity and Readability

To understand our dataset better, we also decided to study how difficult these comments could be interpreted. As we mentioned earlier that our dataset is from Wikipedia Talk Page, we expect users to have more discussion on edits and improvements to articles. In this case, the complexity of their comments should probably be higher than plain chat languages. If a comment has toxic words and strong tones, it might be easier to understand than the content and changes to its associated article. We explored a few readability assessment methods and decided to use Flesch-Kincaid Reading Ease, which is good for all sectors and disciplines, given the wide coverage of Wikipedia topics. The higher the score, the easier the text can be understood. We aggregated all the comments for each category into one composed long text. From *Fig 6*, we can see that our assumptions are partially right. Toxic, severe_toxic, obscene, threat and insult have higher readability scores than identity_hate and non_toxic. However, due to the imbalance in our dataset, the length of the text varied a lot for each category. The mathematical formula of the scores indicates that the sentences that contain a lot of words are more difficult to follow. This also explains why the blue and red lines in our *Fig 6* are moving in opposite directions. If we calculate the readability score of each comment, it will also be tough to have enough sentence count. As a result, we did not include the readability as a feature in our model.

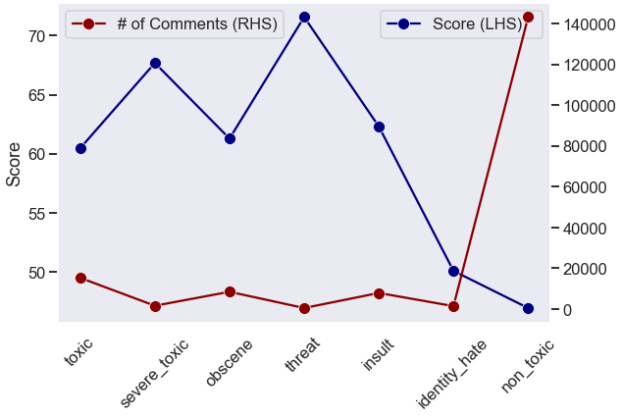

Fig 6. Flesch Kincaid Reading Ease vs the Number of Observations

3.2 Topic Modelling - Latent Dirichlet Allocation (LDA)

In natural language processing, topic modelling is an unsupervised machine learning technique that can scan a set of documents. Then automatically clusters the documents based on its word and phrase patterns. One of the most popular topic modelling techniques is known as Latent Dirichlet Allocation (LDA). LDA is a generative statistical model that allows sets of observations to be explained by unobserved groups that explain why some parts of the data are similar.

To have a better understanding of our data, topic modelling is performed on both datasets.

Kaggle dataset

First, we plot the top 20 most frequent words for unigram, bigram and trigram.

From the *Table 1* we can tell that when the text is split into threes, it contains more toxic/insulting phrases. This could be explained when people tend to repeat those words a few times to express their emotions.

Table 1. top 20 most frequent words for unigram, bigram and trigram

N-GRAM	TOP 20 WORDS
UNI-GRAM	articl, edit, would, use, one, like, pleas, delet, thank, sourc, see, think, make, know, also, time, go, user, people, say

BI-GRAM	Would like, speedi delet, reliabl sourc, http www, fair use, person attack, block edit, edit war, fuck fuck, feel free, pleas stop, nigger nigger, en org, delet artic, edit articl, let know, hate hate, http en, edit summari, ip address
TRI-GRAM	Fuck fuck fuck, nigger nigger nigger, hate hate hate, http en org, en org wiki, hi moron hi, moron hi moron, criterion speedi delet, faggot faggot faggot, lol lol lol, pig pig pig, fat jew fat, jew fat jew, as as as, suck suck suck, shit shit shit, speedi delet articl, four tild automat, fish fish fish, bark bark bark

3.2.1 LDA- Kaggle Dataset

Then we perform the LDA model and set the max_features to be 15,000, min_df to be 3, max_df to be 0.75 and ngram range to be 1, 3. This means the model only kept the 15,000 most occurring words as features, also these words must occur in at least 3 documents and maximum in 75% of the text lines also it will consider all text combinations in unigram, bigram and trigram. Reason being words with low frequency or words that occur in almost every document are usually not a good parameter for classifying as they normally do not provide any unique information about the text lines.

Since latent Dirichlet allocation is a probability model, we will evaluate the model performance with perplexity score and log-likelihood. Perplexity score is used to measure how well the topic model fits the data by computing the average log-likelihood of the train data. Log-likelihood is used to compare the fit of different coefficients to determine how plausible model parameters are in the given data. We chose the result with higher log-likelihood and lower perplexity score.

From the result generated, we can conclude that the text is best split into 10 topics as it has the lowest perplexity score of 2703 and highest log-likelihood of -49139865.

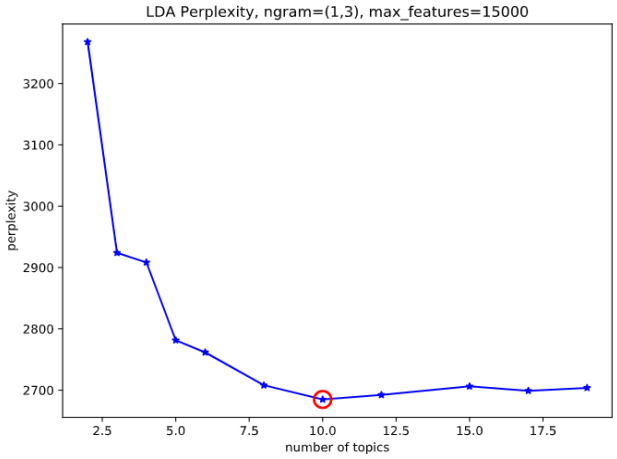

Fig 7. LDA Perplexity Score

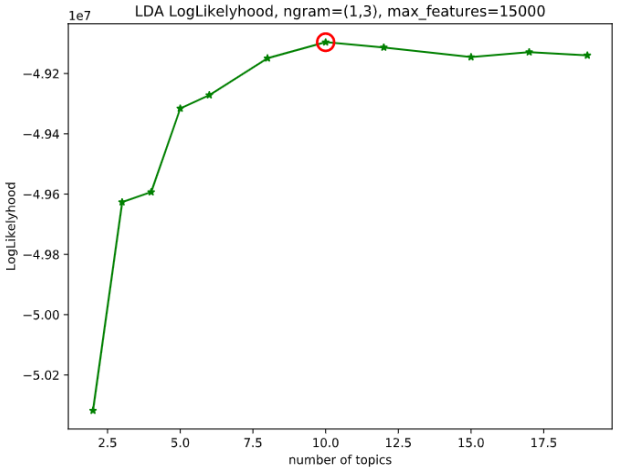

Fig 8. LDA Log-Likelihood

Table 2. The most frequent 15 words for each topics

TOPIC #	TOP 15 FREQUENT WORDS
TOPIC #0	Imag, fuck, use, copyright, fair, upload, fair use, delet, file, jpg, fuck fuck, fuck fuck fuck, tag, bitch, medium
TOPIC #1	would , know, think, like, articl, one, make, time, see, discuss, good, comment, look, want, thing
TOPIC #2	Name, state, year, world, peopl, also, english, nation, language, american, countri, call, first, one, respons
TOPIC #3	Name, state, year, world, peopl, also, english, nation, language, american, countri, call, first, one, respons

TOPIC #4	Block, user, edit, person, delet, attack, get, wiki, admin, care, stop, like, peopl, guy, shit
TOPIC #5	Delet, articl, tag, speedi, speedi delet, notabl, pleas, may, criterion, subject, guidelin, note, delet articl, or wiki, oppos
TOPIC #6	Articl, sourc, use, refer, section, inform, wp, also, remov, need, fact, one, includ, claim, point
TOPIC #7	Edit, pleas, thank, vandal, user, hi, revert, help, block, ip, messag, continu, use, ban, contribut
TOPIC #8	Utc, help, question, style, welcom, edit, ask, 2005, color, use, articl, hope, name, place, date
TOPIC #9	Get, day, go like, one, suck, origin, way, people, lot, life, categori, bit, back, base

Observing the top 15 words from each topics, we can see that topic #0, topic #3, topic #4 is equivalent to the label severe_toxic, identity_hate and threat in our kaggle dataset. As topic #0 contains the most number and emotional negative words. Topic #3 contains hate words that is related to people's identity such as nigger, jew. Topic #4 contains words that threaten to block / delete the post another person has made. This is an example of how we can label our data on our own.

3.2.2 LDA- Scraped Dataset

Then we also performed topic modelling on our scraped dataset with the same parameters except setting the max_features to be 10,000 since there is less word for scraped dataset. The suggested number of topics is 5. (Refer to appendix fig 3 & 4 for the LDA Perplexity Score and LDA LogLikelihood

Observing the top 15 words in the topics, we can see that none of the topics is negative or toxic.

4. Data Preparation

4.1 Data augmentation to address class imbalance issue

As seen from *Fig 1*, the dataset is imbalanced. 89.8% of the comments are non_toxic, which means they have 0 for all labels. It is natural in real-word classification problems, but it is challenging to model training and may result in poor predictive performance for the minority classes. The purpose of this project is to find out toxic comments, thus data augmentation technique is explored to mitigate the effect of imbalance data.

We define all remaining data apart from non_toxic as abnormal_comment, which means they have at least one label to be value 1. Data augmentation is applied to abnormal_comment only. Textblob is adopted to extend training dataset, by translating a comment to an intermediate language (e.g.: French, Chinese) and then translate it back to English using Google's translate API. We assume that the meaning of the comment remains the same but the wording may change a little bit after data augmentation. In the translation process, sleep time is used to avoid too many request errors. It is tested from 0.2 with step of 0.05, which turns out that 0.5 is the best choice for this case. It takes around 6 hours for one round of the translation of abnormal_comment. To save time, it is split to 4 sub datasets and run the data augmentation code on separate colabs, followed by combining the processed sub datasets.

French and Chinese are chosen as the intermediate language to execute the data augmentation for abnormal_comment. After the data augmentation, the abnormal_comment is scaled up to 3 times of its original size. Comparing the processed comment with its original abnormal_comment, we can find the difference between them, which is consistent with our assumption. Taking 2 abnormal_comment for example:

Table 3. Example of text augmentation - words and expressions changed

TRANSFORMATION	TEXT EXAMPLE
ORIGINAL TEXT	Take care not to believe your own bullshit...
EN-FR-EN	Be careful not to believe your own bullshit...
EN-CN-EN	Please be careful not to believe your own nonsense...

Table 4. Example of text augmentation - grammar corrected

TRANSFORMATION	TEXT EXAMPLE
ORIGINAL TEXT	You are gay or antisemmitian?
EN-FR-EN	Are you gay or anti-Semitic?
EN-CN-EN	Are you gay or anti-semitic?

4.2 Feature engineering and feature selection

As discussed in the exploratory data analysis section, there are meaningful features inside the comment texts such as average percentage of capital letters and average count of exclamation marks, which may help us distinguish toxic comments from normal ones. Engineering such features can help pick up a lot of information that TF-IDF and neural networks are not able to capture.

It is not feasible to manually visualize and check all the new features. As a result, we created multiple extra features and applied an algorithm called Boruta to automatically select features. The features we created can be summarized into following four categories:

- **Basic features**: num_unique_words, capital_per_char, punctuation_per_char etc
- **Grammer related**: verb_per_word, adjective_per_word, num_simple_words etc
- **Readability score**: Flesch-Kincaid Reading Ease, Gunning Fog Index etc
- **Sentiment score:** AFINN sentiment analysis and TextBlob sentiment polarity

Boruta works as a wrapper algorithm around Random Forest. It adds randomness to the given data set by creating shuffled copies of all features and iteratively trains a random forest classifier on the extended data set to calculate the feature importance score. It then checks whether a real feature has a higher importance score than the best of its shadow features and constantly removes features which are deemed highly unimportant. In our project, Boruta helped reject 7 features and also generated a box plot of feature importance during the 100 iterations (Fig 9 shows an example box plot of part of our features). We used the information from Boruta to guide feature selection in the model training part.

The feature engineering step is performed before text preprocessing so that we can extract useful information such as percentage of capital letters, number of punctuations etc before they are removed in the cleaning process.

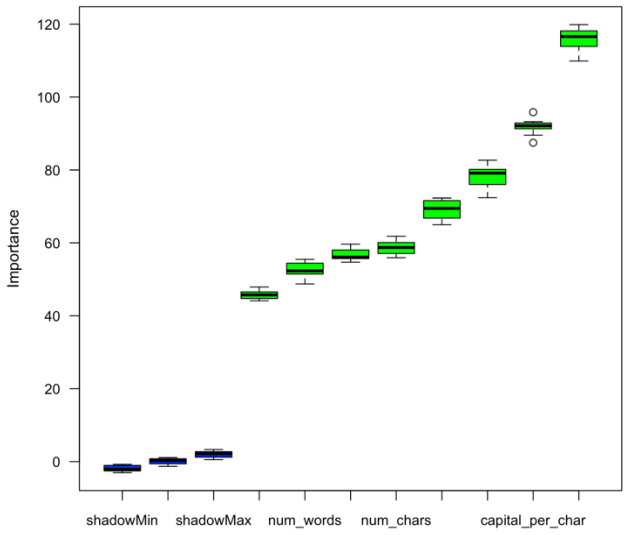

Fig 9. Box plot of feature importance from Boruta algorithm

4.3 Text preprocessing

In order to transform text into a more digestible form for machine learning algorithms, a standard text preprocessing procedure is utilised on the dataset. Different text preprocessing techniques are investigated for modelling, and one or many of the following text preprocessing methods are used for different models, including:

- Togging text to lowercase
- Removal of special characters (except exclamation mark, question mark etc)
- Removal of whitespaces
- Removal of leading and trailing space
- Removal of stop words
- Removal of hyperlinks and emojis
- Replacing all numbers with ‘n’
- Word level fixing:
    - Transforming the shortened form to complete form of selected words or expressions
    - Replacing redundant words with the right format
    - Deleting triple or more consecutive letters (i.e gooood to good)

5. Multi-Label Classification Model Building

In this section, we trained various models to perform the multi-label text classification task. The cleaned dataset is

randomly split in training (90%) and validation (10%) sets. The single models are evaluated using mean column-wise **ROC AUC** (Area under the ROC Curve). In other words, the score is the average of the individual AUCs of each predicted column. After comparing and analyzing the performance of the models, we will select the best model to build our final toxic comment classification system.

5.1 Classic Models

5.1.1 Baseline single models

We trained four single models as our baseline models. The models we explored are LogisticRegression, Multinomial Naive Bayes, LGBM Classifier and Decision Tree. Since we are facing a multi label classification problem, OneVsRest strategy was implemented. OneVsRest treats multi label problems as mutually exclusive and trains different classifiers corresponding to each label and converting this problem into a binary classification problem.

We use the cleaned dataset to train and validate our models. All input comments are transformed into tf-idf vectors. At this stage, the default settings are used for all models and no extra features are added.

Table 5. Summary of baseline model performance

Model	AUC-ROC
Dummy Mode 1	0.5128
Logistic Regression	**0.9802**
Multinomial NB	0.8446
LGBM Classifier	0.9469
Decision Tree	0.7385

From the summary table, we can see that all the baseline models are able to outperform the dummy model (random guesses) by a relatively large margin. Among them, logistic regression achieves the highest AUC ROC score. Thus we decided to add extra features to the logistic regression model and fine-tune it in the next stage to improve its performance.

5.1.2 Fine-tuned model with hand crafted features

For the logistic regression model, we increased the inverse regularization strength C to 4 and stacked six extra features (including percentage of capital letters, percentage of unique words, SMOG score etc.) to the sparse tf-idf matrix. This boosted the model's performance on validation set from 0.9802 to 0.9825. Next we tuned the TfidfVectorizer to include bi-grams and also adjusted parameters such as word frequency cut-off thresholds. The performance of our logistic model was further improved to **0.9840**. We also tried incorporating Naive Bayes log-count ratio with the input tf-idf matrix, following the idea stated in Wang etl.'s paper. But this trick did not work very well for our case.

5.2 Deep learning Models

5.2.1 RNN model with Bidirectional LSTM and GRU layers

The first deep learning model we explored is a relatively standard model taking advantage of RNN and word embeddings. The first layer of our model is an embedding layer where we concatenated pretrained fasttext and glove twitter embeddings together as weights. Following the embedding layer is a SpatialDropout1D layer to randomly drop 50% of the embedding dimensions, which serves as a regularization step. We then stack one Bidirectional CuDNNLSTM layer and one Bidirectional CuDNNGRU layer sequentially. After that, a layer is created by concatenating the previous state with a max pooling layer, an average pooling layer and two hand crafted feature inputs. The final layer is a dense output layer with 6 nodes. The overview of our model's structure can be found in Appendix Fig. 5.

To train and validate the model, we used the cleaned train and validation dataset from our text preprocessing pipeline. The words are then directly tokenized and the comment texts are converted to sequences accordingly. Because of our limited computing resources, a max length is set to truncate (or pad) the input sequences to 100 words.

The RNN model achieved an AUC ROC score of **0.9884** on validation set after 5 epochs of training. Note that the model is able to achieve satisfactory performance even with minimal feature engineering and limited number of training epochs.

5.2.2 Pertained BERT model

BERT is a popular transformer based model that has achieved state-of-the-art results on various NLP tasks. In this project, we also explored the performance of BERT on multilabel text classification tasks. Here we utilized a pretrained BERT model released by Google Research: BERT-Base, Uncased: 12-layer, 768-hidden, 12-heads, 110M parameters. We loaded the pre-trained model, added a linear classification layer on top of the pooled output and then fine-tuned it for our specific classification task.

Similar as before, we take the cleaned train and validation dataset from our text preprocessing pipeline. We applied BertTokenizer to tokenize the words and then converted our data into a format that BERT understands (added the special "CLS" and "SEP" tokens used by BERT to identify

sentence start and end etc.). The max_seq_length was set to 256 to speed up the training process and reduce computational costs.

The model was trained for 4 epochs, but it seems to suffer from overfitting from the third epoch (validation loss kept increasing afterwards). Thus we early stopped at the second epoch, which achieves an AUC ROC score of **0.9890** on the validation set. Through leveraging the power of transfer learning, we were able to obtain a satisfactory model without performing any extra feature engineering.

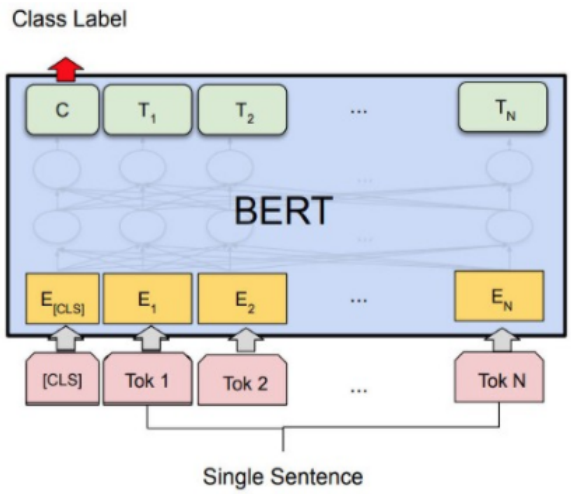

Fig 10. Illustration of BERT structure

5.3 Insights from applying machine learning methods

From applying the various machine learning methods to solve the toxic comment classification problem, we have several key takeaways:

1. Deep learning models are powerful and expressive. Through leveraging pre-trained word embedding and pretrained models, we are able to achieve desirable performance with minimal manual feature engineering and preprocessing.
2. Feature engineering and selection are important. Feature engineering is able to pick up some information missed by tf-idf vectorization. Adding extra features helped boost the performance of our baseline linear models. However, careful feature selection is also critical. We experienced adding destructive features (e.g. TextBlob Sentiment polarity, percentage of big words etc.), which added noises or caused overfitting, and led to performance drop on the validation set.
3. In our business scenario, we need to consider trade-offs when we decide on the final model to deploy. Deep learning models require few domain knowledge and are able to achieve best performances (even with limited training epochs and feature size). But they require long training time and consume a lot of computation power. On the other hand, classic machine learning models such as logistic regression are light-weightt, easy to train and deploy. With careful feature engineering and model tuning, they are able to obtain comparable performance with the deep learning models (0.984 vs 0.989).

In this toxic text classification project, unlike medical or credit default related problems, we are relatively more tolerant with the model's performance. In our business use case, a linear regression model with AUC score of 0.984 can solve most of the problems. In addition, as we plan to deploy our model to detect real time comments on social platforms such as live video rooms, it is crucial for our model to be lightweight and require short inference time. Thus we picked the **improved linear regression model** as our best model.

In the next sections, we applied this best model to the scraped twitter dataset to qualitatively assess its performance on external dataset. We also built a simple dashboard on top of this model to market our solution to business users.

5.4 Future model improvement and extensions

Our model currently only works with English texts. For future work, we plan to extend our model's capability to detect toxic contents in multi-language use cases. Dataset imbalance is also a major concern in this project, we tried augmenting data in the minority classes. We should also try increasing the weight of minority classes, and exploring more proper models to further tackle this data imbalance issue. Last but not least, we trained separate models for each of these six toxic classes when we applied the traditional machine learning techniques. Whereas, as a multi-label problem, there may exist correlation between the classes. To further extend our work, we will also take the relation between classes into consideration.

5.5 Qualitative model assessment on twitter dataset

To further prove our understanding of model training and selection is right. We decided to apply the model on the scraped twitter dataset to test if our model can accurately detect toxic text. Our model will give us a probability percentage that the text falls under each label and we determine if the text is toxic or not by looking if the text has at least one label over 50%. Out of the 10k observations 1632 are labelled as a toxic comment. Base on the word cloud produced from the text labelled as toxic we can see

negative words such as 'fuck', 'shit', 'die', 'idiot', 'kill' are among the most frequent words. This can prove that our model is somewhat accurate in spotting toxic texts.

Fig 11. Word cloud top 20 most frequent words.

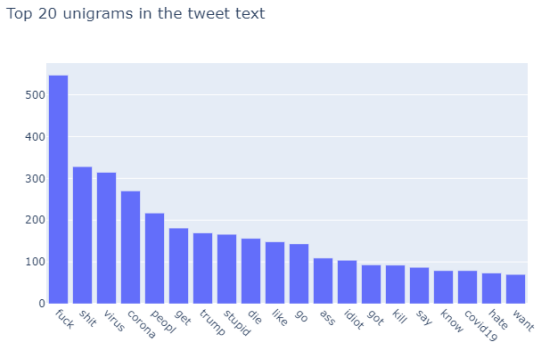

Fig 12. From tweets labelled as toxic.

6. Business Impacts & Applications

6.1 Dashboard Demonstration for Business Users

Our solution for toxic comment detection and classification can be applied in many business cases as what we discussed previously. At the same time, to market our solution to business users, we prepared a simple dashboard using Dash to give them a feel of how our model can be used. The dashboard (see *Fig 13* and *Fig 14*) includes an input box for users to type in whatever they want to test. There will be two outputs being generated after calling our logistic regression model to run a prediction on the input text. The first one is a system detection result, displaying a message that if the comment has been updated successfully or blocked. The second one is a bar chart showing the 0-1 toxicity scores for each of the six toxic categories. Here we defined a rule that if the total score is greater than 2 or at least any two categories' scores are greater than 0.6, the comment cannot be approved by our system. For this rule, different organizations can set up their own guidance. Also, we chose to use our baseline model logistic regression in the dashboard instead of the deep learning model. This was in the consideration of a tradeoff between accuracy and computation power. To run our model in the backend with processing a huge number of comments every day, it may require a lot of effort and depend on our business users' server capability and business scale.

Fig 13. Dashboard response of non-toxic comments

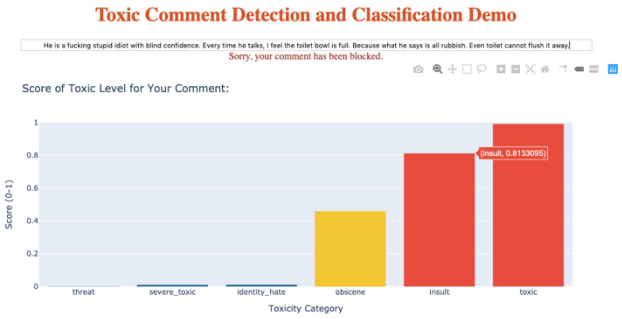

Fig 14. Dashboard response of toxic comments

6.2 Business Scenarios

We recall the purpose of this project is focusing on toxic comments detection and classification. Our final model is able to identify whether the given words or sentences are toxic or not, as well as to categorise them into different levels. Originally it gives the probability of toxicity of 6 categories. They can be further processed to solve the real word problems.

Taking the prosperous social media for example, our model can help to keep the social media platform environment healthy. Nowadays, individuals are exposed to a wealth of information, and the social media platforms are faced with a large amount of comments from instant to instant. It requires a lot of manpower and time for the monitoring team to keep the platform healthy. Things become worse for live video platforms. In the live video

room, the toxic comments are seen by all audiences, the host of the live video room and the admin from the monitoring system. There is no time for the Admin to take any action. With the implementation of our model, the toxic comments can be detected and classified automatically. The model is implemented as a filter between the comment collection and comment broadcasting. All comment input by the user will pass through the filter before sending out to all others. The comment can be seen by the host and audiences only when they meet the judgment criteria from the social media platform. The judgment cretia is customised and up to the social media platform, based on the model results. Meanswhile, for the convenience of the monitoring team, the toxic comments can be highlighted, marked with label or warning sentences in the front end of the monitoring system, a mapping should be constructed between model results and the front end components. An extension of the user case is that Admin can use the model results as one of the parameters to form the punishment rule for the users or the live streaming room. From the perspective of the company, it helps to save manpower and increase the effectiveness at the same time.

Another business scenario is to use the model results as input to another module, so as to increase the performance or enrich the analysis resource of the module. Taking the user portrait module in traditional to-C application for example, our model can act as one attribute to complete the user portrait module. It is quite common for a to-C application to construct the user portrait system, so as to understand users better, provide better service and help the business development team to do precision marketing. With our model implemented to the user portrait system, all comments of a user are passed to the model, a time series of toxic scores are calculated from the model results with customised rules. The time series scores are used as input into the user portrait system.

7. Conclusions

In this project, we aim to build machine learning models which are able to automatically detect toxic comments on social media and live-video streaming platforms. We used a toxic comment dataset from Kaggle to train and validate our models. We also built our own twitter dataset on coronavirus topic to qualitatively assess our model's performance. Various exploratory data analysis techniques such as word cloud plot and LDA were applied to help us better understand the datasets and drew interesting insights from them. We came up with a complete pipeline of data augmentation, feature engineering and text preprocessing to prepare our dataset before the model training process. Four traditional classification models and two deep learning models were explored in our study. We picked the fine tuned logistic regression model as our best model because it is lightweight compared to the deep learning models and gives us a relatively good result in terms of AUC-ROC score. Finally, we prepared a simple dashboard to demonstrate how our solution works and identified two potential business cases as examples for the business owners.

Github link:
https://github.com/Zanyy2k/BT5153_Applied_Machine_Learning_Group18

References

Rainie, L., Anderson, J., & Albright, J. (2019, December 31). The Future of Free Speech, Trolls, Anonymity and Fake News Online. Retrieved from https://www.pewresearch.org/internet/2017/03/29/the-future-of-free-speech-trolls-anonymity-and-fake-news-online/

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. arXiv preprint arXiv:1301.3781.

Pennington, J., Socher, R., & Manning, C. D. (2014, October). Glove: Global vectors for word representation. In Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP) (pp. 1532-1543)

Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.

Wang, S., & Manning, C. D. (2012, July). Baselines and bigrams: Simple, good sentiment and topic classification. In Proceedings of the 50th annual meeting of the association for computational linguistics: Short papers-volume 2 (pp. 90-94). Association for Computational Linguistics.

Kursa, M. B., & Rudnicki, W. R. (2010). Feature selection with the Boruta package. J Stat Softw, 36(11), 1-13.

Appendix – Charts and Tables:

Fig 1. Word Cloud for Kaggle Toxic dataset .

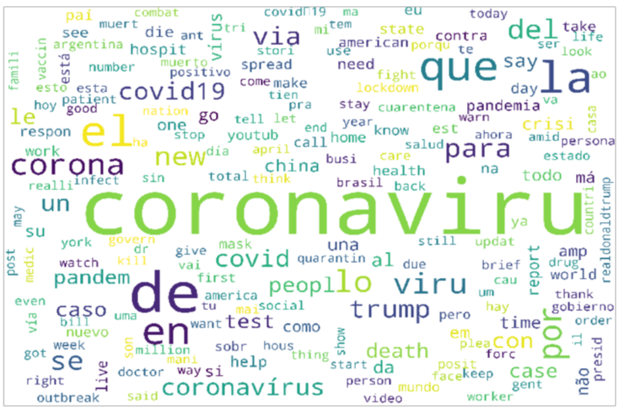

Fig 2. Word Cloud for scraped twitter dataset

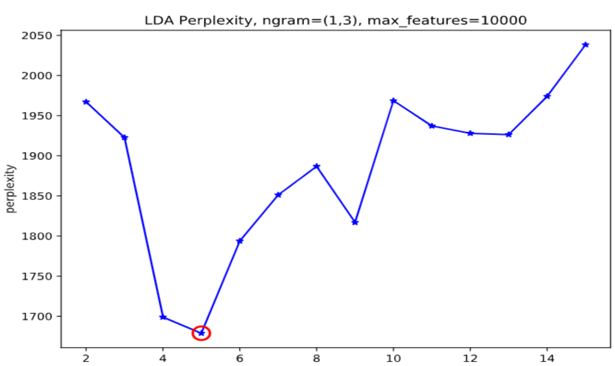

Fig 3. LDA perplexity

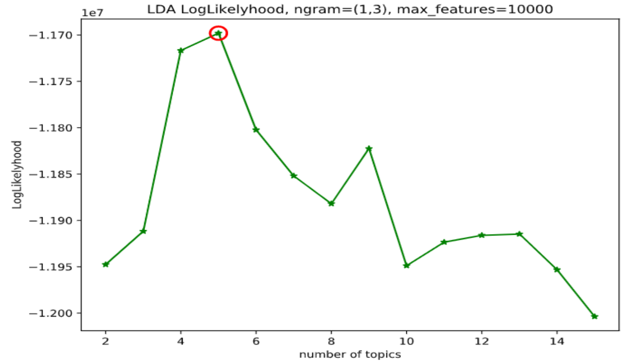

Fig 4. LDA Log-Likelihood

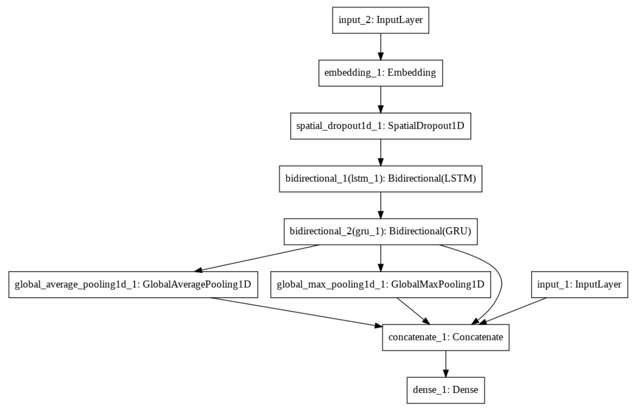

Fig 5. Structure of RNN model

Table 1. LDA topics of scraped twitter dataset

Topic #	Top 15 Frequent Words
Topic #0	de, la, el, en, que, por, los, del, el coronavirus, se, un, con, para, de coronavirus, de la
Topic #1	De , covid, coronavirus, 19, que, covid 19, corona, da, virus, corona virus, não, em, com, eu, para
Topic #2	Trump, via, people, new, pandem, say, get, test, covid19, time, like, go, news, coronavirus pandem, china
Topic #3	Virus, corona, corona virus, death, case, le, covid19, 000, covid-19, stay, coronavirus case, coronavirus death, via, vaccine, covid19 coronavirus
Topic #4	Si, contra, no, crisi, te, contra el, bbc, contra el coronavirus, news, el, la crisi, por, bbc news, la, tiger

Table 2. Example toxic comments detected from twitter

Category	Examples
Toxic	Damn This Coronavirus Got Y'all In a Fucked Up Situation ✨✨
Severe_Toxic	FUCK A BITCH NAMED CORONAVIRUS!!!
Obscene	Part 3 got delayed cause dumb ass Corona virus 😒
Threat	I hope you get the corona virus and die
Insult	Fuck coronavirus and fuck Donald Trump
Identity Hate	And now coronavirus is gay