DBA3803
Group 2
A Singaporean Comment Filter
Project Report

Chuen Liang Shen Kenny | A0166714B
Daniel Lee Wei Sheng | A0167810E
Ng Huong Min, Eugene | A0168344Y
Tan Kai Wei | A0167781R
Tan Quan Hao | A0167827N
Tay Teow Shuen, Rayner | A0170617H

Table of Contents

1 Project Objective

To censor comments on online platforms based on predetermined categories in a uniquely Singaporean context. These censored comments can be segmented into further topics based on affairs being discussed. This model can be utilised for the Media Authority/Regulators and Social Media Websites (e.g. Info-communications Media Development Authority, Twitter, YouTube, Reddit, etc.).

2 Business Problem

The advent of social media has led to many young children being exposed to indecent content on the internet. A study by DQ Institute showed that 54% of Singaporean children between the ages of 8 to 12 are exposed to cyber risks (Kwang, 2018). Equally as important is the rise in the number of comments on Singapore platforms, aimed at raising discontent and anger towards the Government, as well as other members of our community. For instance, Edwin Tong, Senior Minister of State for Law, stated that during Malaysia's recent border conflict with Singapore, there had been a spike in comments that were critical of Singapore, seeking to implant distrust amongst local netizens (Lim, A., 2019). Most insidious about these comments are that they are made to appear as 'general sentiments' of the average Singaporean, when in fact, could simply be the work of falsehood spreaders. Thus, the need for filters and censorships may be extremely pressing to shelter our children as well as the rest of the populace. It is also necessary to analyse which topics the censored comments are revolving on; however, one difficulty will be the grammar and vocabulary of our colloquial language (Singlish), which combines elements of Hokkien, Mandarin, Malay and English, and has its own unique slang and syntax. This may undermine the effectiveness of existing text filtering algorithms, and thus necessitate a need to create one that is more 'uniquely Singaporean' to fit into our local context.

3 Data Understanding

In sourcing for contextualised local comments, web-scraping was done on local forums, news sites and social media (E.g. SammyBoy Forum, HardwareZone Forum, Yahoo News, Twitter, Reddit, etc.). 20,000 individual comments were manually tagged according to the offensive categories: Insulting, Xenophobic, Racist, Anti Government, and Sexual (Refer to Figure 1).

Comments	Insulting	Anti Government	Xenophobic	Racist	Sexual
Singapore those lousy ministers..still called themselves first class	1	1	0	0	0
foreign trash	1	0	1	0	0
Erection coming soon....	0	0	0	0	1

Figure 1: Tagged Training Data

4 Data Preprocessing

We limited the maximum length of the comments to 280 characters, based on the maximum length of a Tweet on Twitter (Perez, 2017). This was to prevent overly lengthy comments which would contain many unnecessary words that possibly could affect our algorithm.

To preprocess our textual data we first turned all characters to lower-case, this was to ensure words like “pap” and “PAP” would be treated the same. Using Regular Expressions we converted the short form of words to their full form to ensure uniformity (e.g. can’t to cannot, would’ve to would have. We also had to take into consideration the colloquial nicknames used and ensure they all referred to the same entity (e.g. converting “gahment” to “government”). Moreover, we also had to remove foreign characters and find patterns to remove links, such as "https://", and unwanted strings such as, “This image has been resized”. We removed stop words (e.g. is, from, the) from our comments which are common words which appear frequently across many comments and also do not assist with our information retrieval, adding little semantic value. Lastly, the words are then lemmatised to obtain their root word by analysing the morphology of the word based on its Part of Speech tag (e.g. verb, adjective, adverb). For example, a word such as “cars” would be converted to “car” so that “cars” and “car” can be detected as the same feature.

We leveraged on N-grams, which are combining “N” number of a continuous sequence of words to form a single feature. This is as a combined word could have a different interpretation from each word, standing alone. Using Bigrams (two words) we were able to combine words such as “Cold Storage” and “Heart Attack” which may have a very different meaning from “Cold” and “Storage” as well as “Heart” and “Attack”. However, we also utilised Unigrams (one word).

4.1 Term Frequency-Inverse Document Frequency (TF-IDF)

The goal of using Term Frequency-Inverse Document Frequency (TF-IDF) instead of the raw frequencies of occurrence of a token in a given comment is to scale down the impact of tokens that occur very frequently across many comments in a given corpus and that are hence empirically less informative than features that occur in a small fraction of the training comments.

The term frequency (TF) contains the frequency of the word over the total number of words in the comment. The inverse document frequency (IDF) calculates the total number of comments over the number of comments that contain the word. By inverting the fraction, it rewards words that occur rarely in the whole corpus of comments. A natural logarithm is put in place to dampen the effect of obtaining an extremely large IDF where the corpus contains many comments. By multiplying the TF and IDF together we simultaneously reward words that occur frequently in a specified comment but punish the words that occur frequently across all the comments.

$$w_{i,j} = tf_{i,j} \times \log\left(\frac{N}{df_i}\right)$$

Figure 2: Term Frequency-Inverse Document Frequency Equation (Tripathi, M., 2018)

Our output would be a TF-IDF Document-Term Matrix. This would contain in our case, rows containing each of the 20,000 comments in our training data and the columns containing all of our unique features. Thus each cell represents the TF-IDF value of a feature for a specific comment.

	government	foreign	minister
0	NaN	0.498299	NaN
1	0.701315	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	0.238277
5	NaN	NaN	0.459175

Figure 3: Term Frequency-Inverse Document-Term Matrix

4.2 Normalisation of TF-IDF

We added 1 to the IDF in case of an extreme situation where the feature appeared in all the comments. In such a situation log(n/n) would be 0 and an IDF of 0 would lead to a TF-IDF of 0. We would not want such a commonly occurring feature to have a weight of 0. 1 is also added to the denominator for cases whereby the feature is new and not found in the dictionary. Thus the document frequency would be 0 and we cannot divide by 0. Hence, 1 is added to numerator as well, this is to prevent the denominator from getting larger than the numerator which may cause the IDF to be negative, which we do not want. Moreover, for the TF we chose to use the pure frequency count of the feature in the comment rather than dividing that number by the total number of terms in the comment. Our reasoning is if a comment contained an extremely vulgar word, its importance should not be lessened just because the comment contains a lot of features. Hence a vulgar feature in a short comment should have the same importance as the same feature in a lengthy comment.

$$\text{IDF} = \log\frac{N+1}{DF+1} + 1$$

Figure 4: Normalisation of Inverse Document Frequency

5 Models

5.1 Baseline Model (Simple Logistic Regression)

There was no normalisation and preprocessing for the data used in the baseline model. A Bag of Words (BoW) Document-Term matrix was created using the count of the frequency of a unique unigram (single word) in all 20,000 comments. The rows represented each comment and the columns represented each unique unigram. We fitted the x variable as the counts of unigrams for each comment and the y variable as either “1” or “0”, indicating whether the comment contains the label. With this, five regression models representing each label, were obtained with a single BoW Document-Term Matrix.

	government	foreign	minister
0	NaN	1.0	NaN
1	1.0	NaN	NaN
2	NaN	NaN	NaN
3	NaN	NaN	NaN
4	NaN	NaN	1.0
5	NaN	NaN	2.0

Figure 5: Bag of Words Document-Term Matrix

5.2 Bayesian Logistic Regression

5.2.1 Likelihood

For each label, the TF-IDF weights of each feature, given the comments containing that respective label, was summed. The total sum of TF-IDF values for each feature was then divided by the total number of comments that contained that label. Hence, we were able to retrieve a probability of the TF-IDF value of each feature given the comment contains such a label. For example, we were able to find the likelihood of the TF-IDF value of the feature "stupid" given the comment is "Insulting". Carrying out the same procedure but vice versa we were able to obtain the likelihood of the TF-IDF value of each feature given the comment does not contain such a label.

P(TF-IDF of feature|comment contains label) - likelihood of TF-IDF of each feature given comment contains label for each label
P(TF-IDF of feature|comment does not contain label) - likelihood of TF-IDF of each feature given comment does not contain label for each label

5.2.2 Ratio

We log divide the likelihood given comment that has the label against the likelihood given the comment that does not have the label to obtain a Bayesian ratio. The log is used to dampen the effect of the division as if the numerator and denominator differ largely the resulting ratio can have either an extremely large value or one close to 0. Hence it carries out a subtraction instead between the log of both likelihoods.

$$\log \frac{P(TF-IDF\ of\ feature|comment\ contains\ label)}{P(TF-IDF\ of\ feature|comment\ does\ not\ contain\ label)}$$

Figure 6: Bayesian Ratio

5.2.3 Multiplication of Ratio with TF-IDF Document-Term Matrix

For each feature, we multiplied its TF-IDF ratio for each label against the feature's TF-IDF value in the Document-Term Matrix through a dot product. Henceforth, we obtained five different ratio adjusted TF-IDF Document-Term Matrices, each representing a label. This multiplication helps to alter the TF-IDF values. For example, "stupid" may have an extremely high ratio, due to having appeared much more frequently in "insulting" comments as compared to "non-insulting" comments. By multiplying this ratio to the TF-IDF value, we can rightfully increase its value in the "insulting"

Document-Term Matrix to a much larger number. Thus, signifying that the feature has much more importance in an insulting comment.

5.2.4 Model

Thereafter we fit the ratio adjusted Document-Term matrix for the respective label as the x variable and the tagged labels (1 or 0) as the y variable into 5 separate binary logistic regression models for each of the 5 labels.

5.2.5 Reason for not using Multinomial Logistic Regression

We decided not to apply this method because we had 5 separate labels that were not mutually exclusive and thus could occur together (e.g. "The Government is idiotic", is Insulting and Anti Government). Thus, we could not proceed with Multinomial Logistic Regression as it is suitable in a situation with a single label but with multiple categorical outcomes. We did have the option of creating different combinations of the 5 labels, but that would lead to $2^5 = 32$ different combinations. This would have been an extremely excessive number of classes, with many being irrelevant. Furthermore, this would not have been in line with our business problem as we aimed to censor comments that contained any of the 5 labels. However, by using the 32 classes, it seems to be more focused towards finding the specific label combination of the comment. In essence, this business problem required a multi-label classification model rather than a multi-class classification model.

5.2.6 Constraints of Bayesian Logistic Regression

Due to having five Bayesian Logistic Regression separate models, the five labels were not able to be taken into consideration together to determine if a comment should be censored. For example, a case may arise where a comment is deemed clean as it is a borderline case for each of the 5 labels. However, if we did take into consideration the probabilities for each of the 5 labels combined the comment may be considered censorable.

5.3 Feature Engineering

5.3.1 Model

Feature Engineering is a commonly used model that relies heavily on domain knowledge of the dataset to produce features. The creation of each feature requires to some extent, guesswork, to produce significant and impactful features. Our desired goal is to be able to discover a relationship between the features and each of the 5 labels for classification.

To start off, we simplified the model by screening through the dataset and identifying characteristics of all comments. Subsequently, we decided on 23 features, with each feature being a count of the number of a particular word that exists in the comment. The number 23 which we stopped at is purely arbitrary; and if increased will enhance the model's robustness.

The 23 features that we decided on were 'atb', 'bbfa', 'amdk', 'nnp', 'chiobu', 'sinkies', 'cunt', 'cpf', 'prc', 'pap', 'pappies', 'ang mo', 'dog', 'knn', 'cb', 'fuck', 'fk', 'pussy', 'boobs', 'xmm', 'fark', 'ah neh', and 'oppies'.

Each of the features were chosen based on their distinctiveness in the Singlish language, as well as the meanings behind them. Most are acronyms that exist within the Singlish dictionary, of which a majority have a negative connotations when used. These are ideal for the purpose of our project in detecting content to be filtered. The features were then appended to the train dataset, with each value corresponding to the count of the feature within the particular comment.

	Comments	Insulting	Anti Government	Xenophobic	Racist	Sexual	atb	bbfa	amdk	nnp	...	knn	cb	fuck	fk	pussy	boobs	xmm	fark	ah neh	oppies
1570	KNN some uncle riding motorised wheel chair an...	1	0	0	0	0	0	0	0	0	...	3	0	0	0	0	0	0	0	0	0
1571	Sinkies going crazy Both loony deserves each o...	1	0	0	0	0	0	0	0	0	...	0	0	1	0	0	0	0	0	0	0
1572	This type of old fucks are hot air only. Chall...	1	0	0	0	0	0	0	0	0	...	0	0	2	0	0	0	0	0	0	0
1573	The old man has every right to enjoy his music...	1	0	0	0	1	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0
1574	Wah what a threat.	0	0	0	0	0	0	0	0	0	...	0	0	0	0	0	0	0	0	0	0

Figure 7: Each comment goes through the same counting for every feature

The subsequent step is to determine correlation between each feature (23) and each of the labels (5). This was done by correlating, for example, the column of 'knn' against the column for the label 'Insulting'.

5.3.2 Point Biserial Correlation

For the correlation matrix, Point Biserial Correlation was used to compute the correlation between the feature and label. The output returned is a correlation value that lies between -1 and +1, as well as a p-value to determine if the correlation value is significant.

The Point Biserial Correlation equation (Statistics How To, 2016) is as follows:

$$r_{pb} = \frac{M_1 - M_0}{s_n}\sqrt{pq}$$

Where $\mathbf{M_1}$ = mean of entire label group that were assigned '1's
$\mathbf{M_0}$ = mean of entire label group that were assigned '0's
$\mathbf{S_n}$ = standard deviation over all binaries for that particular label
P = proportion of cases of '1' in the entire label group
Q = proportion of cases of '0' in the entire label group

The correlation values are filtered according to the following 3 basic conditions:

1. **Significant P-value (<0.05).** As we are identifying features that are significant, we only look at features that give us a correlation values with significant p-values.
2. **Positive Correlation Values.** As the purpose of our filter is to detect comments positively related to our labels, we are not interested in features that are negatively correlated to our labels.
3. **Correlation value (>set threshold of 0.01).** We are interested in features with a strong correlation with our labels. As such, we filtered out values that are less than 0.01. This filter determines the level of strictness we would like to achieve and a higher threshold would filter out more words.

To better visualize the relationships between each feature and each category, a heatmap of the correlation values is then plotted. The more intense hues of blue is a signal that the feature on the y-axis correlates strongly with the category on the x-axis.

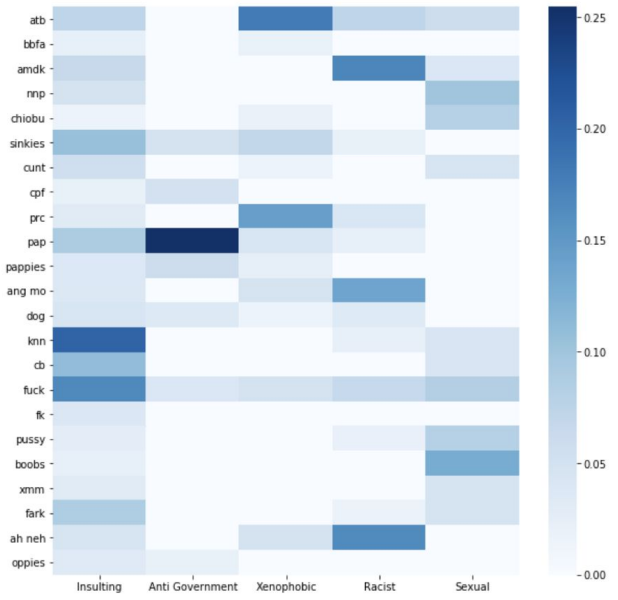

Figure 8. Heatmap visualization of correlation values between all features and all categories

5.3.3 Reason for not using Pearson's Correlation

$$r = \frac{n(\Sigma xy) - (\Sigma x)(\Sigma y)}{\sqrt{[\, n\Sigma x^2 - (\Sigma x)^2 \,][\, n\Sigma y^2 - (\Sigma y)^2 \,]}}$$

The traditional method for computing correlation between two variables would be to turn to Pearson's Correlation. However, despite the ease of inputting Pearson's, we opted for a more accurate computation due to the fact that the label variable was a binary variable. As Pearson's correlation is suitable for only 2 continuous variables, a different mathematical solution was required.

5.3.4 Significance of Feature Engineering

To utilize feature engineering as a model, the process of appending the count of the features can be replicated on any new comment. By obtaining a 1 x 23 matrix, with each column being the count of the same features used in the train set, matrix multiplication with the correlation matrix (23 x 5) can be applied to obtain a 1 x 5 matrix output.

	Insulting	Anti Government	Xenophobic	Racist	Sexual
0	0.927552	0.295106	0.1274	0.09041	0.525394

Figure 9. 1 x 5 matrix output with a computed score for each category

However, it has to be duly noted that the final score output in the 1 x 5 matrix is not a scale of probability or degree to which the comment violates the category. Instead, the score simply translates the presence of features that falls into the 5 labels, relative to each other. Thus, we cannot compare the final scores between comments. However, within a single comment comparisons can be made. In this case the comment can be evaluated as 'more insulting than it is xenophobic'.

5.3.5 Constraints of Feature Engineering

A key constraint of feature engineering is the human decision in determining the exact features to be used. As the features were selectively chosen before application in the model, it is imperative that the features picked were impactful.

Furthermore, our features that were chosen were simplistic in nature. The features could be further complicated by moving beyond count of words, but taking into account ratios or other statistical aspects as well.

However, as we recognised the downfalls of feature engineering, it was not used as a conclusive model to determine the classification. Instead, feature engineering was used to complement the Bayesian Logistic Regression. By implementing the count of the features as a weight, the 23 features were appended to the TF-IDF dataframe, essentially acting as another feature in the ratio adjusted TF-IDF Document-Term matrix that was fitted as the x variable in the logistic model. Refer to Appendix D for the combined TF-IDF Document-Term Matrix.

The enhanced model is more robust and accurate as the new TF-IDF Document-Term matrix takes into consideration features that were manually picked under human influence and decision. However, time complexity of the model increased exponentially as a result of a denser TF-IDF Document-Term matrix.

5.4. Random Forest Model

5.4.1 Model

Random Forest is a learning method for classification that constructs multiple decision trees based on samples in the dataset and takes the averages of the results out. This merging of multiple decision trees aims is done so to obtain a more accurate and stable prediction. The features extracted using TF-IDF text representation will be utilized to classify the training data texts. Similar to the Bayesian Logistic Model, we decompose the multi-label problem into 5 independent binary-class problems such that there are 5 different random forest models, one for each label. The number of tree parameter is set to 1000 trees so as to increase performance.

5.4.2 Feature Bagging

Random forest utilises this technique to increase accuracy and decrease correlations among the decision trees on average. Feature bagging operates by selecting at random a subset of the feature dimensions at each split in the growth of individual decision tree. However it allows random forest to deliberately avoid strong predictive features that result in similar splits in the individual trees (and thus increase correlation). For example, in a standard bagging procedure, if a particular feature is strong in predicting the response value then it will be selected for many trees. Hence a standard bagging procedure can be quite correlated. Random forests avoid this by deliberately leaving out those strong features in many of the grown trees. When it comes to making a prediction, the random forest takes an average of all the individual decision tree estimates and output the average score.

5.4.3 Gini Impurity Metric

Random forest model uses the Gini Impurity Metric as a measure of impurity. The algorithm minimises the weighted average gini impurity within each node as we move down the tree. Gini impurity for a set of instances with J classes is a measure of variance in the tree. The higher the variance, the more mis-classification there is and thus, lower values indicates better classification in the tree. p_i denotes the probability of an instance with class i being chosen i.e. fraction of instances with class i in the set.

$$Gini(E) = 1 - \sum_{i=1}^{J} (p_i)^2$$

5.4.4 Reason for not using Classification Trees

Classification Trees are high variance estimators. Hence, an addition of a small number of extra training data points can dramatically affect the prediction performance of a learned tree, although the training data did not change drastically. In contrast, as established above, random forest seeks to reduce overfitting.

5.4.5 Algorithm

Training Stage:

1. Randomly select x number of features from all the features
2. Among the x features, calculate the node d using the best split point.
3. Split the node into daughter nodes based on the best split.
4. The forest is built by repeating steps 1 to 3 for n times to create n number of decision trees.

Prediction Stage:

1. Takes the test features and use the rules of each randomly created decision tree to predict the outcome and stores the predicted outcome (target)
2. Calculate the votes for each predicted target.
3. Consider the highest voted predicted target as the final prediction from the random forest algorithm.

5.4.6 Visualisation of Decision Trees in a Random Forest

For a preliminary attempt at visualising a decision tree in a random forest, we used a very small and thus, slightly skewed sample set containing 7 comments for training. Refer to Figure 10 for the comments and Figure 11 for the features extracted from these 7 comments.

Comments	Insulting	Anti Government	Xenophobic	Racist	Sexual
FUCK FOREIGN TALENT	1	0	1	0	0
Singapore malays are still stupid. Thought they are smarter than their neighbouring cousins.	1	0	1	1	0
Jiak liao bee govt got into this mess and only know recommend sueing option, go and sell to more magnimous country to accept them Why bo lum par to help these hiv	1	1	0	0	0
Wah i fapped liaoOrange jiuce kam out aj	0	0	0	0	1
Pump full tankPs: who wants kkj pic, when i got it i forward to ah gui	0	0	0	0	1
Got jav actress almost want to piak liao you now then cum	0	0	0	0	1
fap fap fap fap fap	0	0	0	0	1

Figure 10. Comments for Sample Visualisation

['accept', 'actress', 'ah', 'aj', 'almost', 'and', 'are', 'bee', 'bo', 'country', 'cousins', 'cum', 'fap', 'fapped', 'foreign', 'forward', 'fuck', 'full', 'go', 'got', 'govt', 'gui', 'help', 'hiv', 'into', 'it', 'jav', 'jiak', 'jiuce', 'kam', 'kkj', 'know', 'liao', 'liaoorange', 'lum', 'magnimous', 'malays', 'mess', 'more', 'neighbouring', 'now', 'only', 'option', 'out', 'par', 'piak', 'pic', 'pump', 'recommend', 'sell', 'singapore', 'smarter', 'still', 'stupid', 'sueing', 'talent', 'tankps', 'than', 'their', 'them', 'then', 'these', 'they', 'this', 'thought', 'to', 'wah', 'want', 'wants', 'when', 'who', 'why', 'you']

Figure 11. Features Extracted for Sample Visualisation

For the sample visualisation of the trees, Random Forest model was performed for all 5 labels with number of trees parameter set as 500. Refer to Figure 12 for visualisation of some of the sample trees.

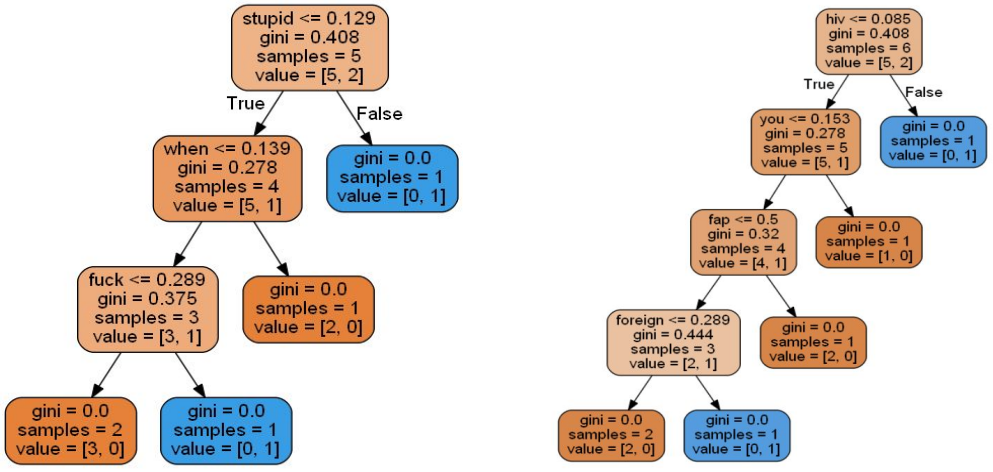

Figure 12a. Insulting Label Sample Trees

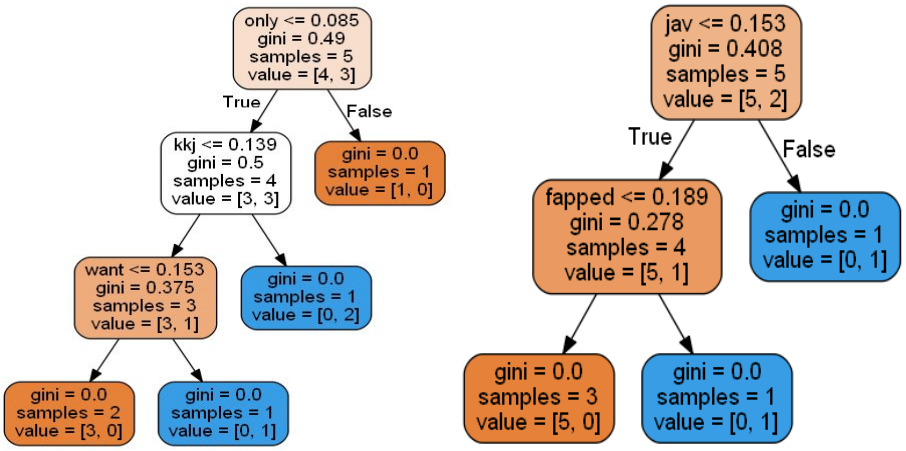

Figure 12b. Sexual Label Sample Trees

In each leaf node, the gini represents the gini impurity score and the sample denotes the number of distinct data points (comments in our case) in each tree. The value denotes the total number of comments, including repeated comments. [*x*,*y*] signifies that *x* number of data points that belong to the left daughter node and *y* number of data points belonging to right daughter node.

The number of sample and value in each node can differ due to bootstrapping, in which it is a random sampling of a dataset with replacement leading to occurrence of repeated data points.

5.4.7 Constraints of Random Forest

Due to the complexity of random forest, it requires a lot of computational resources and is also less intuitive. The prediction process using random forest is also extremely time-consuming compared to the other algorithms.

As we are decomposing the multi-label problem into multiple independent binary classification models, one for each label, we are not accounting for any correlations between the five different labels. This is also a constraint that the Bayesian Logistics Regression faces.

5.5 Multi-Label k Nearest Neighbours (ML-kNN) Model

ML-kNN is an adapted algorithm for multi-label classification (Zhang & Zhou, 2007). It is an extension of the traditional kNN algorithm based on the Maximum-a-Posteriori (MAP) principle.

The MAP estimate is an estimate of an unknown quantity θ, which maximises the posterior distribution. Value of θ that is most probable given the observed data denoted by D and prior probabilities will be chosen.

$$\hat{\theta}_{MAP} = \arg\max_{\theta} P(D|\theta)P(\theta)$$

5.5.1 Reason for not using traditional K Nearest Neighbours Model

We selected the ML-kNN model over the traditional kNN model due to it being designed for multi-label learning. An intuitive approach to handling this multi-label problem is to decompose into 5 independent binary class kNN models. However, as mentioned previously, this will forsake the correlations between the different labels; thus an adapted algorithm is adopted instead.

5.5.2 Algorithm

The crux of the ML-kNN algorithm is that a test instance's labels is dependent on the number of nearest neighbours having the same labels as the test instance. For a test instance x with an unknown label set $L(x) \subseteq L$, where L denotes all 5 offensive labels categories, ML-kNN will first identify the k nearest neighbours in the training data set based on Euclidean distance. Then z, the number of neighbours belonging to each label, will be counted. z is a variable ranging from 0 to k.

The MAP principle is applied to determine the label set for this test instance x. The prior probabilities listed below are estimated from our training data set by ML-kNN. The posterior probability of $li \in L(x)$ is as follow and li represents each single label:

$$P(li \in L(x)|z) = \frac{P(z|li \in L(x)) \cdot P(li \in L(x))}{P(z)}$$

For each label, the algorithm builds a classifier based on the following rule:

$$hi(x) \begin{cases} 1 & P(li \in L(x)|z) > P(li \notin L(x)|z) \\ 0 & P(li \notin L(x)|z) > P(li \in L(x)|z) \end{cases}$$

If $P(li \in L(x)|z) > P(li \notin L(x)|z)$, then $hi(x) = 1$ and thus there is confidence that label li is in test instance x's real label set. If $hi(x) = 0$, then label li is not in x's real label set.

We deployed our model and performed parameter tuning based on cross validation scoring and attained an optimal parameter of k=3.

5.5.3 Constraints of ML-kNN

Even though ML-kNN algorithm is supposed to be adapted for multi-label classification, it may not be a good application for our case context. This is because TF-IDF has high dimensionality but the distance used in the nearest neighbours that ML-kNN algorithm is Euclidean Distance. Euclidean Distance is not a strong distance metric to be used for comparisons in high dimensional data. Instead, measures such as Cosine Similarity may be more relevant. Cosine Similarity ignores the length of the comments as it ignores scales and can function well with high dimensional data. Furthermore, as ML-kNN uses MAP for classification, estimation of the priors and posterior probabilities can be skewed due to imbalances between instances with a certain label and those without. In addition, various literatures have cited that ML-kNN algorithm is not a complete multi-label classification solution and tends toward binary relevance i.e. decomposing into multiple independent binary learning tasks, similar to the Bayesian Logistics Regression and Random Forest models (Chiang, Lo & Lin, 2012; Liu & Cao, 2015).

6 Topic Modelling

To further improve our project, we explored topic modelling as well. Topic modelling allows us to extract topics from a large collection of texts to find out the issues that Singaporeans are expressing their views on. Government agencies or entities dealing with censorship can then identify topics that are being repeatedly censored and these comments can be used to reflect the sentiments of the public. The two topic models we explored are the Latent Semantic Analysis model and the Latent Dirichlet Allocation model.

6.1 Latent Semantic Analysis (LSA)

Latent Semantic Analysis is a method for extracting topics from a large collection of texts by utilising the dimensionality reduction property of matrices (Landauer, Thomas et. al., 1998). After removing words that hold little or no meaning, the remaining words are converted into a BoW Document-Term matrix containing the count of the frequency of the features for each comment. Dimensionality reduction can be performed using truncated Singular Value Decomposition (SVD) where the Document-Term matrix is factorised into the Document-Topic matrix U, diagonal matrix S containing the singular values of M and the Term-Topic matrix V. The parameter t refers to the t most significant number of topics that we want to identify.

$$A \approx U_t S_t V_t^T$$

We first determined the t most prominent topics based on the number of topics which give us the highest coherence score. These coherence scores are computed based on the "c_v" coherence score.

c_v retrieves the co-occurrence counts for words in the dataset. These counts are used to compute the normalized pointwise mutual information (NMPI) of every word to every other word, resulting in vectors. The average of the cosine similarity between the vector of each word and the vectors of all other words gives the c_v coherence score (Pradhan, Ligaj, et. al., 2016). Comparing models with different number of topics, the model with 16 topics gave us the highest coherence score of 0.40. The words associated with the 16 most prominent topics can be found in Appendix E1. Based on our results computed using LSA, the topics which are commonly discussed are forums containing sexual content as well as MRT disruptions.

6.2 Latent Dirichlet Allocation (LDA)

Latent Dirichlet Allocation is a generative probabilistic model for collections of discrete data (Blei D., Ng A., Jordan M., 2003).

LDA assumes that documents are produced from a mixture of topics and these topics comprises of a certain probability distribution of terms. From there, LDA aims to extract the terms that would form these topics (Bansal S., 2016). Given the comments dataset, we specify the number of topics K we need. The LDA algorithm first forms a Document-Term matrix (D*W) which is then broken down into Document-Topic (D*K) and Topic-Term (K*W) matrices.

	W1	W2	W3	W4
D1	0	2	1	3
D2	1	4	0	0
D3	0	2	3	1
D4	1	1	3	0

*Figure 13a. Example Document-Term matrix, with 4 documents and 4 distinct terms (4*4). Numbers represent occurences of each term in each document.*

	K1	K2	K3	K4
D1	1	0	0	1
D2	1	1	0	0
D3	1	0	0	1
D4	1	0	1	0

	W1	W2	W3	W4
K1	0	1	1	1
K2	1	1	1	0
K3	1	0	0	1
K4	1	1	0	0

Figure 13b. The Document-Topic matrix and Topic-Term matrix formed from the above example. Assuming 4 topics are specified, the matrices comprise 4 documents with 4 topics, and 4 topics with 4 terms respectively. Numbers in the Document-Topic matrix represent occurences of each topic in each document. Numbers in the Topic-Term matrix represent assignment of each term to each topic.

For each document, each term is first randomly assigned a topic. The algorithm then iterates through every term w within every document d. For each term, it tries to assign a new topic k to it, calculating probabilities p_1 and p_2 for each.

- p_1: The proportion of terms in document d that are currently assigned to topic k

$$p_1 = P(k \mid d)$$

- p_2: The proportion of assignments to topic k over all documents that come from this term w

$$p_2 = P(w \mid k)$$

The two probabilities p_1 and p_2 are multiplied to get the topic-to-term probability, and if there is an improvement in this computed probability for the new topic k, it is assigned to the term. In doing so, LDA assumes that all term-to-topic assignments other than the current one are correct. This continues until a specified number of iterations, or when a steady state is reached when the topic-to-term probabilities for each term cannot be improved further. Then, each and every term would have been assigned with the most probable topic.

Similarly to what we did with regards to the LSA model, after iterating LDA models topics within the range of t=4 and t=20, the model with 16 topics gave us the highest coherence score of 0.33. The words associated with the 16 most prominent topics can be found in Appendix E2. Based on our results computed using LDA, the prominent topics include law enforcement, Singaporeans' dissatisfaction with the country, as well as dating and relationships.

6.3 Limitations of Topic Models

Words with little semantic value were given importance in the topics, thus we had to manually remove such words from the dictionary by extending the list of stop words. Furthermore, the meaning of the associated words is interpreted by the person extracting the topics; an identical set of words might result in different identification of topics by another person.

7 Model Evaluation

7.1 Score Comparison

For each different model type, we came out with five different sub-models, one to predict each category[1]. For each and every sub-model, 5-fold cross validation was performed from which we calculated the standard deviation; we also plotted the Receiver Operating Characteristic (ROC) curve, and found the area-under-curve (AUC). Thereafter, the mean of all five sub-models was calculated to obtain the standard deviation and AUC values for the models as a whole.

Finally, to determine the run time score, we also ran the same test string across all five models and measured their execution times using Python's *time* library.

The final performance of each model, with respect to both scores, can be seen in Figure 14.

	Simple Logistic Regression (Baseline)	Bayesian Logistic Regression	Combined Bayesian Logistic Regression with Feature Engineering	Random Forest	Multi-label k Nearest Neighbours
AUC	0.7923	0.8628	0.8629	0.8188	0.5053
Standard Deviation	0.0536	0.0381	0.0372	0.0588	0.0043
Test Comment Run Time	1.56 sec	2.05 sec	318.22 sec	781.87 sec	14.17 sec

Figure 14. Performance across all models

To select the best-performing model, we decided to choose the models that performed the best in terms of both AUC and standard deviation. As such, we eliminated the baseline and Random Forest models because they returned relatively lower AUC and higher standard deviation. Then, the Multi-label k Nearest Neighbours model was also taken out because even though it performed the best in terms of standard deviation, it was also the worst performing with respect to AUC. This poor performance can be attributed to firstly its use of Euclidean Distance as a distance metric, which is not suitable for high-dimensional data and secondly, its imbalanced estimation of priors and posterior probabilities that is heavily reliant on the training data.

[1] See Data Understanding, page 3

Among the remaining two models, Bayesian Logistic Regression and Combined Bayesian Logistic Regression with Feature Engineering, the AUC scores are very close. We applied the principle of parsimony, sacrificing a relatively small amount of performance for a model that is significantly simpler, as shown by the differences in run time in Figure 14. Computing time is a particularly important point for consideration, given our already massive amount of training data.

Hence, Bayesian Logistic Regression is our model choice for deployment.

7.2 Other Limitations and Constraints

7.2.1 Imbalance in Training Data

The majority of our data was scraped from websites that contained vulgar content such as HardwareZone Forum and SammyBoy Forum. A seemingly innocent comment such as "PAP is good" could possibly be detected as Anti Government, due to the sheer frequency of the feature "PAP" being found amongst comments tagged Anti Government, leading to it having a very high ratio adjusted TF-IDF value from the insulting Document-Term matrix, and in turn a very high probability under the Anti Government label. This could be seen as a situation of overfitting, where the feature "PAP" gathers a negative semantic from the training dataset, rather than acting as a neutral feature.

A solution would be to have much more balanced and larger dataset with a mix of comments highlighting these key features in a positive light as well. However, due to the time constraint of our project and the tediousness of manual tagging, we had to focus a larger proportion of our training data on vulgar comments so as to generate more censorable training examples.

7.2.2 Sarcasm & Other Literary Techniques

Due to a large proportion of sarcastic comments on Singaporean social media, this may be an issue that leads to inaccurate predictions. Often, these sarcastic comments are phrased very similarly to comments with good intentions, with the only difference being the surrounding context. When manually tagging comments, we may be able to detect the sarcasm, but that is only a partial solution. The algorithm may not be able to detect sarcasm the same way that humans do.

Other literary techniques in the English language such as irony, pretense or paradoxical comments were also not accounted for in our analysis due to the complexity of such comments. However, bringing context into the equation is probably the best approach out there to getting more reliable results. For example, this can be done by taking into consideration the features of the author of the comment, features of the forum thread and features of users who have "liked" or responded to the comment.

7.2.3 Adaptability of Training Model

As our models were developed based upon current and past comments made, the model is limited in keeping up with new lingo or Singlish terms. The drawback is that our models are unable to evolve to account for new colloquial terms used. To be able to consistently sieve out negative comments, a solution would be to constantly re-train and update the training dataset with the latest comments and terms. This would then increase the adaptability and reliability of the classifications of our models.

8 Model Deployment

8.a Flask Web App

As a further application of our selected model - Bayesian Logistic Regression, we created a web app using the Flask framework for Python, which serves as a prototype of our comment filter in the public domain.

The app is operated by two main components, *main.py* which handles the text input and result display, and *model.py* which runs our selected Bayesian Logistic Regression model on the training data. When opened, users are provided a text field to enter their comments for checking. Text entered is then posted as a string to *model.py* which predicts the categories of the text entry based on what it has learnt from the training data. Following which, users are alerted whether their comment could be "banned" should the model predict any combination of Insulting, Anti Government, Xenophobic, Racist, or Sexual categories; otherwise, the web app returns "Clean" if the model predicts none of the above mentioned categories.

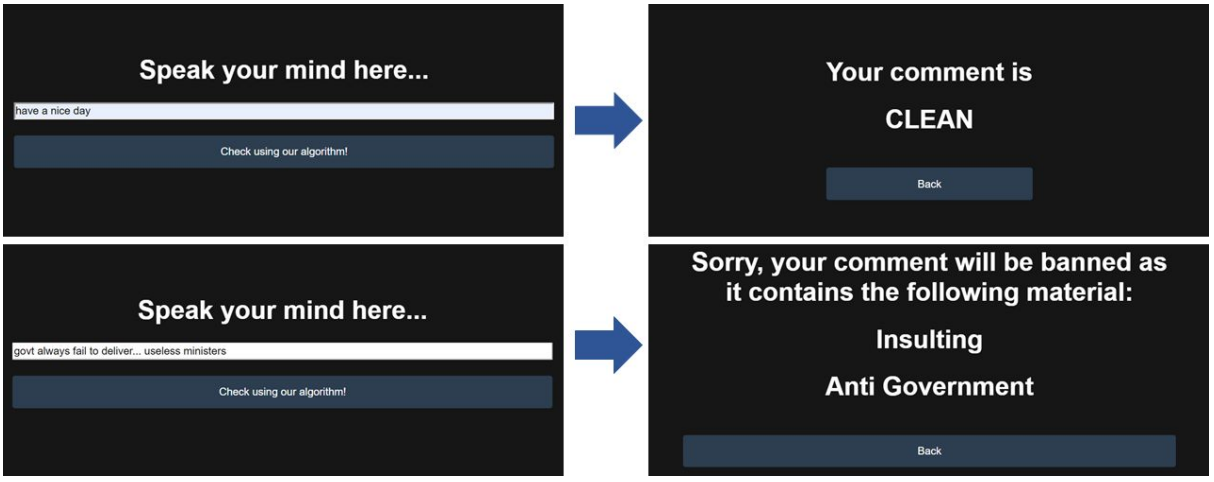

Figure 15. An example of the web app in action. If "have a nice day" is entered, the model aptly does not predict any of the bannable categories. On the other hand, if Insulting and Anti-Government sentiment is predicted, this is picked up by the model and shown.

The web app can be deployed on your local machine as well. A guide on how to deploy using the files is attached in Appendix A of this report.

8.b Future Expansion and Application

Our current model provides a basic framework to analyse comments from various online social and communication platforms used by Singaporeans, which can be easily expanded to achieve further purposes and applications for other platforms.

For instance, with the upcoming General Elections, more labels can be created to measure whether there is Pro or Anti-PAP sentiment, and Pro or Anti-Opposition sentiment. The results of the models can serve as an early indicator for polling results. Topic modelling can be utilised to identify widely discussed issues in Singapore leading up to the elections.

Our models can be easily scalable and applied to other colloquial languages with similar syntax and vocabulary; a prime example being our neighbouring Malaysia's Manglish. In this aspect, we could create a Manglish dictionary and filter comments in their unique context.

Lastly, we could look to Neural Network models such as the Long Short-Term Memory (LSTM) model to further supplement and improve our sentiment analysis accuracy.

9 Appendix

Appendix A: Flask Web App (Deployment)

How to deploy the web app on your local machine (Ganry, T., 2018):

1. Create a folder to store all the files. We will use DBA3803 for this example.
2. Create two folders, *static* and *templates*, and arrange the files in the folder hierarchy as shown below:
    - DBA3803
        - static
            - *style.csv*
        - templates
            - *input.html*
            - *result.html*
        - *main.py*
        - *model.py*
        - *requirements.txt*
3. Open Anaconda Prompt.
4. Change the directory accordingly to access your DBA3803 folder.
    - `cd C:/Users/User/DBA3803`
5. Change the directory to access the training data *train.csv* on *main.py*.
6. Install the system libraries virtualenv and virtualenvwrapper-win using PIP.
    - `pip install virtualenvwrapper-win`
    - `pip install virtualenv`
7. Create and activate the virtual environment for the web app.
    - `virtualenv venv`
8. Install the required libraries as mentioned in the *requirements.txt*.
    - `pip install -r requirements.txt`
9. Activate the virtual environment.
    - `call venv\Scripts\activate.bat`
10. Activate the *main.py* script. It will take quite some time for the web app and model to get up and running, but once done, it will be deployed on your machine's local host. Instructions will be returned to you on Anaconda Prompt once it is ready.
    - `python main.py`

Appendix B: Flask Web App Code Snippets

```
@app.route('/')
def index():
    return render_template(
        'input.html',
        data=[])
@app.route("/result" , methods=['GET', 'POST'])
def result():
    data = []
    error = None
    select = request.form.get("comment")
    resp = censor(select)
    pp(resp)
    if resp:
        data = resp
    return render_template(
        'result.html',
        data=data,
        error=error)
```

A snippet of the *main.py* code which handles the posting of user-entered data into *model.py* for processing.

```
# Censorship
def censor(comment):
    attributes = []
    test_str = clean_text([comment])
    test_com = pd.Series(test_str)
    test_vec = vec.transform(pd.Series(test_com))
    test_pred = np.zeros((len(test_com), len(label_cols)))
    for i, j in enumerate(label_cols):
        m,r = test_mdl(train[j])
        test_pred[:,i] = m.predict_proba(test_vec.multiply(r))[:,1]
        if test_pred[:,i] >= 0.5:
            attributes.append(j)
    if len(attributes) != 0:
        attributes.insert(0, "Sorry, your comment will be banned as it contains the following material:")
        return(list(attributes))
    else:
        attributes.append('CLEAN')
        attributes.insert(0, 'Your comment is')
        return(list(attributes))
```

A snippet of the *model.py* code. The function shown here takes the input string, makes its prediction, and outputs the results accordingly.

Appendix C1:
23 x 5 matrix storing all Point Biserial Correlation values for Features Engineering

	Insulting	Anti Government	Xenophobic	Racist	Sexual
atb	0.072640	0.000000	0.180260	0.073172	0.059406
bbfa	0.023500	0.000000	0.020208	0.000000	0.000000
amdk	0.064862	0.000000	0.000000	0.169275	0.039963
nnp	0.047777	0.000000	0.000000	0.000000	0.100029
chiobu	0.016879	0.000000	0.019456	0.000000	0.080712
sinkies	0.105564	0.048368	0.070204	0.022562	0.000000
cunt	0.055034	0.000000	0.015603	0.000000	0.045327
cpf	0.022761	0.050745	0.000000	0.000000	0.000000
prc	0.031182	0.000000	0.143816	0.042170	0.000000
pap	0.090002	0.254674	0.044424	0.023839	0.000000
pappies	0.037421	0.059944	0.024501	0.000000	0.000000
ang mo	0.038161	0.000000	0.045769	0.137671	0.000000
dog	0.044416	0.035965	0.016351	0.033927	0.000000
knn	0.202712	0.000000	0.000000	0.023284	0.043702
cb	0.110418	0.000000	0.000000	0.000000	0.043030
fuck	0.165769	0.040432	0.047918	0.066571	0.084177
fk	0.040409	0.000000	0.000000	0.000000	0.000000
pussy	0.027167	0.000000	0.000000	0.021327	0.081063
boobs	0.022973	0.000000	0.000000	0.000000	0.129303
xmm	0.031896	0.000000	0.000000	0.000000	0.046668
fark	0.086944	0.000000	0.000000	0.018332	0.046377
ah neh	0.045036	0.000000	0.047817	0.164612	0.000000
oppies	0.032938	0.022855	0.000000	0.000000	0.000000

Appendix C2:
23 x 5 matrix storing P-Values from computing Point Biserial Correlations

	Insulting	Anti Government	Xenophobic	Racist	Sexual
atb	8.16813e-25	0.0767095	1.11446e-145	3.71701e-25	4.16406e-17
bbfa	0.000888409	0.206544	0.00426405	0.678347	0.0717782
amdk	4.23109e-20	0.154899	0.0552025	1.8791e-128	1.57206e-08
nnp	1.3789e-11	0.920283	0.540738	0.543555	1.20329e-45
chiobu	0.01698	0.649068	0.00593152	0.11344	2.88323e-30
sinkies	1.15368e-50	7.709e-12	2.78754e-23	0.00141805	0.0650391
cunt	6.78485e-15	0.385081	0.0273458	0.405437	1.42621e-10
cpf	0.00128579	6.9418e-13	0.123845	0.482291	0.152447
prc	1.031e-05	0.45199	6.82169e-93	2.43106e-09	0.0507516
pap	2.98939e-37	1.40096e-293	3.27495e-10	0.000747311	0.00187505
pappies	1.19887e-07	2.16633e-17	0.00052974	0.647829	0.219344
ang mo	6.72174e-08	0.946428	9.43667e-11	3.28907e-85	0.586583
dog	3.29788e-10	3.62638e-07	0.0207557	1.59387e-06	0.141566
knn	1.70374e-184	0.245029	0.837316	0.000990941	6.29084e-10
cb	2.73287e-55	0.168536	0.348758	0.713617	1.14483e-09
fuck	3.40208e-123	1.06553e-08	1.20073e-11	4.32657e-21	8.78367e-33
fk	1.08674e-08	0.788999	0.558497	0.561234	0.393308
pussy	0.000121804	0.460963	0.628929	0.00255977	1.62049e-30
boobs	0.00115766	0.255909	0.681393	0.459616	2.63756e-75
xmm	6.43135e-06	0.27218	0.471703	0.601802	4.02719e-11
fark	7.21808e-35	0.218537	0.149737	0.00952554	5.31718e-11
ah neh	1.86776e-10	0.440351	1.32564e-11	1.741e-121	0.106148
oppies	3.17572e-06	0.00122774	0.645533	0.647829	0.219344

Appendix D: Appending 23 features to the TF-IDF matrix

benefit	beng	beng ah	bengs	beo	beret	...	oppies	capitals	num_exclamation_marks	num_smilies
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	17.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	18.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	25.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	5.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	1.0	NaN	NaN
NaN	NaN	NaN	NaN	NaN	NaN	...	NaN	26.0	4.0	NaN

Appendix E1: LSA Topic Modelling

Topics Generated

```
[(0,
  '-0.554*"train" + -0.364*"time" + -0.287*"fault" + -0.163*"nsl" + -0.147*"delay" + -0.126*"travel" + -0.113*"look" + -0.111*"go" + -0.096*"knn" + -0.090*"people"'),
 (1,
  '0.473*"train" + -0.291*"look" + 0.234*"fault" + -0.228*"knn" + -0.154*"go" + -0.139*"time" + 0.138*"nsl" + -0.128*"people" + 0.127*"delay" + -0.119*"even"'),
 (2,
  '-0.936*"knn" + 0.197*"time" + -0.087*"train" + -0.075*"cb" + 0.073*"look" + -0.062*"lao" + 0.048*"lol" + 0.042*"even" + 0.042*"last" + -0.041*"wong"'),
 (3,
  '0.728*"time" + -0.568*"look" + -0.191*"train" + 0.118*"knn" + -0.086*"delay" + 0.081*"last" + 0.074*"travel" + -0.070*"nsl" + -0.055*"people" + 0.051*"long"'),
 (4,
  '-0.728*"look" + -0.421*"time" + 0.143*"go" + 0.127*"people" + 0.103*"cannot" + 0.101*"day" + 0.100*"ask" + 0.091*"even" + 0.086*"use" + 0.085*"guy"'),
 (5,
  '0.914*"fap" + 0.384*"pew" + 0.058*"look" + 0.054*"ky" + 0.035*"time" + -0.025*"go" + 0.024*"thread" + -0.023*"people" + -0.016*"even" + -0.016*"pay"'),
 (6,
  '-0.590*"ah" + -0.491*"song" + -0.346*"lai" + 0.207*"people" + -0.200*"bo" + -0.143*"day" + -0.133*"mu" + -0.123*"lemon" + -0.094*"gui" + -0.094*"chin"'),
 (7,
  '0.911*"go" + -0.153*"people" + -0.143*"guy" + -0.116*"even" + -0.103*"cannot" + -0.088*"driver" + 0.068*"back" + -0.062*"ask" + -0.059*"post" + -0.056*"well"'),
 (8,
  '0.462*"day" + 0.367*"use" + -0.311*"people" + 0.234*"5" + 0.216*"vote" + -0.170*"ah" + 0.166*"3" + 0.147*"pcwx" + 0.128*"stupid" + 0.127*"ether"'),
 (9,
  '0.721*"day" + -0.342*"use" + -0.212*"vote" + 0.202*"lol" + -0.149*"5" + 0.121*"work" + -0.114*"pcwx" + -0.107*"stupid" + -0.106*"even" + -0.097*"ether"'),
 (10,
  '-0.624*"people" + 0.382*"lol" + 0.293*"guy" + 0.277*"post" + 0.208*"thread" + -0.158*"day" + -0.136*"even" + 0.123*"driver" + 0.106*"ask" + 0.102*"new"'),
 (11,
  '0.583*"lol" + -0.349*"guy" + 0.321*"people" + 0.225*"driver" + 0.176*"ask" + -0.175*"girl" + -0.173*"buy" + -0.170*"even" + 0.142*"post" + 0.141*"thread"'),
 (12,
  '-0.384*"thread" + 0.380*"driver" + -0.379*"people" + 0.337*"even" + -0.271*"post" + -0.252*"guy" + 0.218*"cannot" + -0.209*"new" + 0.178*"ask" + -0.096*"well"'),
 (13,
  '0.503*"lol" + 0.442*"even" + -0.429*"driver" + -0.381*"ask" + 0.219*"cannot" + -0.090*"video" + -0.088*"post" + -0.080*"day" + 0.075*"thread" + -0.070*"son"'),
 (14,
  '-0.638*"guy" + 0.317*"post" + 0.308*"thread" + 0.235*"new" + -0.185*"lol" + 0.162*"even" + 0.150*"buy" + 0.144*"cannot" + -0.103*"people" + -0.089*"day"'),
 (15,
  '-0.359*"even" + -0.329*"post" + 0.307*"n" + -0.257*"guy" + 0.200*"u" + -0.183*"cannot" + -0.180*"thread" + 0.177*"lol" + -0.173*"day" + -0.172*"driver"')]
```

Appendix E2: LDA Topic Modelling

Topics Generated

```
[(0,
  '0.158*"go" + 0.121*"look" + 0.079*"back" + 0.069*"love" + 0.060*"actually" + 0.057*"always" + 0.056*"get" + 0.045*"long" + 0.045*"lot"
  + 0.044*"story"'),
 (1,
  '0.179*"time" + 0.132*"day" + 0.073*"first" + 0.065*"quite" + 0.050*"place" + 0.049*"high" + 0.040*"must" + 0.035*"night" + 0.029*"especially"
 + 0.023*"become"'),
 (2,
  '0.181*"give" + 0.130*"new" + 0.080*"last" + 0.069*"well" + 0.061*"week" + 0.058*"year" + 0.053*"month" + 0.051*"next" + 0.022*"write" +
  0.014*"block"'),
 (3,
  '0.117*"find" + 0.077*"singaporean" + 0.068*"anything" + 0.044*"police" + 0.039*"woman" + 0.039*"idea" + 0.036*"government" + 0.035*"course" +
 0.035*"taste" + 0.032*"law"'),
 (4,
  '0.169*"thank" + 0.089*"may" + 0.066*"already" + 0.057*"old" + 0.056*"show" + 0.046*"better" + 0.042*"watch + 0.039*"level" + 0.035*"eye" +
  0.023*"pass"'),
 (5,
  '0.168*"people" + 0.123*"would" + 0.094*"money" + 0.068*"help" + 0.040*"burn" + 0.039*"good" + 0.031*"issue + 0.028*"report" + 0.027*"public"
  + 0.025*"allow"'),
 (6,
  '0.152*"singapore" + 0.087*"bad" + 0.082*"pay" + 0.059*"run" + 0.051*"learn" + 0.045*"enough" + 0.037*"agree" + 0.034*"pm" + 0.032*"country" +
  0.032*"lose"'),
 (7,
  '0.110*"try" + 0.081*"man" + 0.077*"s" + 0.067*"tell" + 0.066*"guy" + 0.058*"come" + 0.049*"probably" + 0.038*"life" + 0.032*"change" +
  0.032*"date"'),
 (8,
  '0.140*"ask" + 0.075*"someone" + 0.074*"lol" + 0.061*"happen" + 0.059*"buy" + 0.058*"maybe" + 0.050*"video" + 0.039*"read" + 0.039*"wait" +
  0.033*"nothing"'),
 (9,
  '0.084*"live" + 0.079*"could" + 0.058*"let" + 0.050*"family" + 0.046*"sign" + 0.041*"stop" + 0.040*"case" + 0.038*"hope" + 0.037*"offer" +
  0.034*"head"'),
 (10,
  '0.187*"work" + 0.157*"fuck" + 0.058*"area" + 0.053*"instead" + 0.045*"need" + 0.027*"house" + 0.022*"later" + 0.021*"picture" + 0.020*"whole
  + 0.017*"treat"'),
 (11,
  '0.191*"not" + 0.108*"be" + 0.096*"even" + 0.093*"do" + 0.054*"sure" + 0.043*"mean" + 0.041*"pretty" + 0.041*"lah" + 0.038*"something" +
  0.021*"water"'),
 (12,
  '0.103*"take" + 0.088*"job" + 0.087*"keep" + 0.055*"wrong" + 0.048*"end" + 0.047*"paper" + 0.035*"person" + 0.034*"spend" + 0.027*"die" +
  0.027*"safe"'),
 (13,
  '0.070*"much" + 0.062*"thing" + 0.055*"friend" + 0.044*"think" + 0.041*"post" + 0.032*"hear" + 0.031*"seem" + 0.030*"guess" + 0.026*"start" +
  0.025*"today"'),
 (14,
  '0.114*"say" + 0.097*"use" + 0.085*"never" + 0.080*"feel" + 0.078*"way" + 0.064*"see" + 0.050*"girl" + 0.045*"wonder" + 0.044*"kind" +
  0.032*"far"'),
 (15,
  '0.093*"right" + 0.082*"make" + 0.077*"big" + 0.069*"call" + 0.053*"chinese" + 0.044*"open" + 0.042*"happy" + 0.041*"free" + 0.039*"check"
  + 0.031*"white"')]
```

10 References

1. Bansal, S. (2016, August 24). Beginners Guide to Topic Modeling in Python. Retrieved April 19, 2019 from https://www.analyticsvidhya.com/blog/2016/08/beginners-guide-to-topic-modeling-in-python/
2. Blei, D., Ng, A., & Jordan M. (2003). Latent Dirichlet Allocation. Journal of Machine Learning Research 3. 993-1022.
3. Chiang, T., Lo, H., & Lin, S. (2012). A Ranking-based KNN Approach for Multi-Label Classification. ACML.
4. Ganry, T. (2018, November 6). How to build a web app using Python's Flask and Google App Engine. Retrieved April 19, 2019 from https://medium.freecodecamp.org/how-to-build-a-web-app-using-pythons-flask-and-google-app-engine-52b1bb82b221
5. Kwang, K. (2018, March 01). 54% of Singaporean children exposed to at least 1 cyber risk: Study. Retrieved April 19, 2019, from https://www.channelnewsasia.com/news/technology/54-of-singaporean-children-exposed-to-at-least-1-cyber-risk-9932554
6. Landauer, Thomas, et. al. (1998) An Introduction to Latent Semantic Analysis. Discourse Processes, 25, 259-284.
7. Lim, A. (2019, February 12). Parliament: 'Curious' spike in online comments critical of S'pore during dispute with Malaysia, says Edwin Tong. Retrieved April 19, 2019, from https://www.straitstimes.com/politics/parliament-curious-spike-in-online-comments-critical-of-spore-during-dispute-with-malaysia
8. Liu, C., & Cao, L. (2015). A Coupled k-Nearest Neighbor Algorithm for Multi-label Classification. Advances in Knowledge Discovery and Data Mining Lecture Notes in Computer Science, 176-187. doi:10.1007/978-3-319-18038-0_14
9. Perez, S. (2017, November 07). Twitter officially expands its character count to 280 starting today. Retrieved April 19, 2019, from https://techcrunch.com/2017/11/07/twitter-officially-expands-its-character-count-to-280-starting-today/
10. Pradhan, Ligaj, et. al. (2016) Towards Extracting Coherent User Concerns and Their Hierarchical Organization from User Reviews. 2016 IEEE 17th International Conference on Information Reuse and Integration (IRI).
11. Statistics How To. (2016, July 2). Point-Biserial Correlation & Biserial Correlation: Definition, Examples. Retrieved from Statistics How To: https://www.statisticshowto.datasciencecentral.com/point-biserial-correlation/
12. Tripathi, M. (2018, June 06). How to process textual data using TF-IDF in Python. Retrieved April 19, 2019, from https://medium.freecodecamp.org/how-to-process-textual-data-using-tf-idf-in-python-cd2bbc0a94a3
13. Zhang, M., & Zhou, Z. (2007). ML-KNN: A lazy learning approach to multi-label learning. Pattern Recognition, 40(7), 2038-2048. doi:10.1016/j.patcog.2006.12.019