Text Categorization Project Report

Suen Ao Xiang (A0113842M)

1. Introduction

This report describes the methodologies and processes undertaken for the text categorization problem hosted as a Kaggle competition.

2. Dataset

2.1 Exploratory Data Analysis

The dataset provided consists of comments posted on two Q&A websites and are classified into a total of 16 different topics.

Preliminary exploration on the training dataset has found that the number of comments per topic are highly imbalanced. As shown on **Figure 1** below, the number of comments in topic 14 is exceedingly high, at 483,982 comments, whereas the second highest is only 83,464 comments belonging to topic 13. There are several minority topics classes with less than 10,000 comments, notably topic 6 with only 2,328 comments. This indicates that some processing steps is required to solve the imbalanced issue, which would be described in Section 2.6.

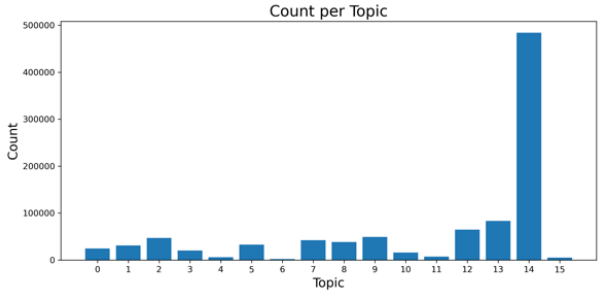

Figure 1: Comment Count per Topic

2.2 Feature Engineering (Part I)

The feature engineering process is conducted mainly in two parts. First part is being conducted before preprocessing the dataset, as there could potentially be relevant features before they are removed in the preprocessing step, such as counting number of special characters. It should be noted that such features are extracted per comment, in other words, there are no aggregation operations to avoid data leakage.

Table 1 below shows the various features and a short description indicating the thought process.

Feature	Description
symbol_count	Number of special characters per comment
number_count	Number of digits per comment
mathexp_count	Number of mathematical expressions per comment, indicated by '$$' prefix
word_count	Number of words per comment
char_count	Number of characters per comment
sentence_count	Number of sentences per comment
avg_word_length	Average length of word per comment
avg_sent_length	Average length of sentences per comment
sentiment	Sentiment per comment
ner_col	Number of different entities identified per comment

Table 1: Feature Engineering (Part I)

2.3 Data pre-processing

Following Part I of feature engineering, the data is pre-processed to standardize as input into the various modelling algorithms. Generic words such as stopwords, digits, and special characters are removed, and words are further reduced to its root form using stemming. Finally, words with less than two characters are removed as the majority of them are not meaningful.

Table 2 below indicates all the pre-processing steps undertaken.

Steps	Description
remove_accented_char	Standardize characters to ASCII format
remove_weblinks	Remove weblinks
remove_non_alpha	Remove special characters and digits
lowercase	Standardize texts to lowercase
remove_stopwords	Remove generic words
stemming	Standardize texts to their root form
remove_short_words	Remove texts shorter than 3 characters. (Majority are meaningless)

Table 2: Data Pre-processing

Figure 2 below is a summary of the pre-processed texts in a form of a word cloud.

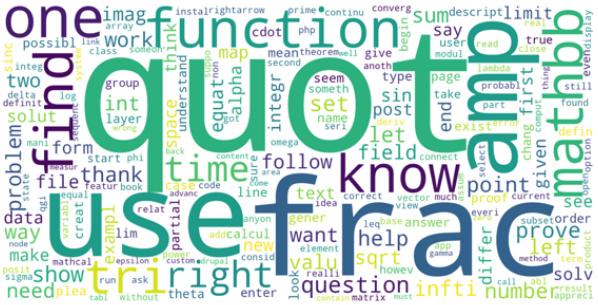

Figure 2: Word Cloud of Pre-processed Texts

2.4 Feature Engineering (Part II)

As this is a multi-class classification problem, whereby comments belong to 16 different classes of topics, it is hypothesized that each comment has a topic distribution that corresponds to its labeled topic class. In other words, we could employ topic modelling techniques to obtain the probability of each comment belonging to each topic and integrate with the classification model.

Latent Dirichlet Allocation (LDA) is utilized for this process. LDA is an unsupervised learning technique that assumes each document has a mix of topics and each topic has a mix of words. Through an iterative process, LDA improves the assignment of topics to words through Gibbs sampling.

As LDA model requires the number of topics to be known beforehand, it is generally derived from domain knowledge or tuned as a hyperparameter. For the case of this project, while it is intuitive to use 16 as the total number of topics, it is observed that a higher number of topics yield better classification results. This could be due to the fact that there are slight overlaps between the comments belonging to different topic classes. Therefore, 20 topics were used in the final model.

Although LDA is an unsupervised learning technique, the purpose of using LDA in this project is mainly for feature engineering. Each comment would have 20 new features comprising of different probabilities of belonging to each topic, which are then used to train the supervised classification model. Therefore, to avoid data leakage, the LDA model is only applied on the training set after splitting 25% as the validation set. In order to obtain a balanced representation of each topic, bootstrap

sampling with replacement is conducted on the training set to obtain 10,000 comments for each topic classes.

2.5 Term Frequency – Inverse Document Frequency (TF-IDF)

In order to convert text data to numerical input for modelling, TF-IDF vectorizer is adopted as a numerical representation of the text corpus. In addition to counting the frequencies of words appearing in each document, TF-IDF also inversely weights the tokens by document frequency. In other words, words that appear in many documents are weighted lower as it is not as meaningful to differentiate the classes. An n-gram range of 1-3 is also used to capture potentially useful word phrases.

2.6 Class Rebalancing

As mentioned in Section 2.1, the topic classes are highly imbalanced, therefore, class rebalancing techniques are utilized to improve the representation of the minority classes. First, as briefly mentioned in Section 2.4, 25% of the training set is split and reserved as a validation set. The split is conducted in a stratified fashion in order to retain the same proportion of majority and minority classes in the validation set.

Next, on 75% of the training set, both undersampling and oversampling techniques are conducted. The majority classes that exceed 150,000 comments (only class 14 in this case) are undersampled to 150,000 comments. This is to avoid oversampling the minority classes to a large extent thereby increasing bias and the size of the data. Classes that have less than 50,000 comments are oversampled to 50,000 using SMOTE (Synthetic Minority Over-Sampling Technique). This approach is observed to yield slightly better results and much faster training time than purely oversampling minority classes.

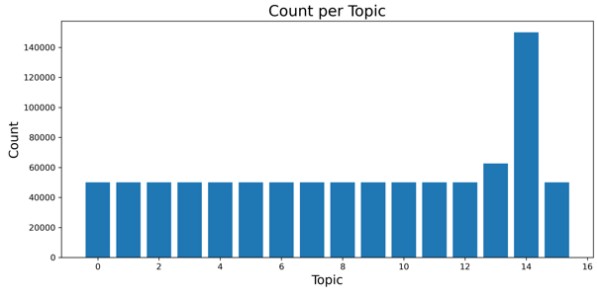

Figure 3: Comment Count per Topic after Rebalancing

2.7 Variance Reduction

As a final data processing step, a dimension reduction technique is used to remove least useful features. Although TF-IDF vectorizer has a feature reduction hyperparameter where words that occur too often and too infrequent can be removed, this step can act as a different method of feature reduction and is mainly for reducing least useful manually engineered features.

The intuition behind the variance reduction technique is that features with very low variance do not contribute meaningfully for topic classification. In other words, features that have the same (or very similar) values for all samples are less meaningful and therefore removed. For that reason, the threshold chosen is very low. Based on observation, threshold of 0.000001 reduces the number of features to approximately half and works well on the validation set.

3. Models and Results

Two models are employed for this project – Multinomial Naïve Bayes and Convolution Neural Networks (CNN) for text. As both models have similar performances and are fundamentally different types of models, one being Bayes Rule with conditional independence assumption, and the latter being a deep learning model, an ensemble technique is suitable to combine both models, and as observed, even outperformed both models individually.

3.1 Multinomial Naïve Bayes

The Multinomial Naïve Bayes classifier is suitable for this use case as it is computationally efficient to be used in large datasets due to the conditional independence assumption. The macro F1-score calculated on the validation set is at 0.807.

3.2 Convolution Neural Networks (CNN)

A CNN model is also used as an alternative to Multinomial Naïve Bayes. The model architecture consists of an embedding layer with a dimension of 50, followed by three 1-Dimensional CNN layers in parallel with 2, 3 and 4 filter sizes respectively, this represents 2-grams, 3-grams and 4-grams. A max pooling layer is used to summarize the respective feature maps, which are concatenated and subsequently input into to a fully connected layer with softmax and an output dimension of 16 which corresponds to the number of topic classes. A detailed architecture plot of this model is attached in the appendix section of this report.

The input length to the CNN model is fixed at 300 words, where longer comments would be shortened by taking the first 300 words and shorter comments would be padded with zeros. In order to mitigate the issue of imbalanced classes, each class is being weighted such that minority classes are weighted heavier, causing the model to "pay more attention" to samples from the minority classes.

Using the same training and validation set as the Multinomial Naïve Bayes model, the CNN model achieved a macro F1-score of 0.814 on the validation set.

3.3 Ensemble

As observed, both Multinomial Naïve Bayes and CNN models performed similarly well on the validation set. A common technique of improvement would be to aggregate both models through ensemble techniques. Soft voting is used in this case where the predicted probabilities of both models are combined and the highest probability for the specific topic class would then be the predicted class. The best weight ratio to combine both models is iteratively tested on the validation set and found to be 0.5 which mean equal weightage contribution.

The macro F1-score on the validation set using this ensemble technique is 0.842, which is an improvement as compared to both models trained separately.

4. Conclusion

In conclusion, it is shown that using ensemble techniques could yield improvements as compared to fitting both Multinomial Naïve Bayes and CNN models separately. However, in spite of that, performance on the test set was considerably lower than it is on the validation set. It could possibly be due to comments that are very short after preprocessing (i.e. 1-3 words), for which there are insufficient information for the models to correctly classify the topics. Also, there is a possibility that the distribution of the test set is different from that of the train set, especially since Multinomial Naïve Bayes utilizes prior probabilities for predictions.

Appendix

CNN Model Architecture