Coursework CMT207 – Text Categorisation **DRAFT DATE: 21/ 04 / 22**

## Abstract

#

#

#

#

## Introduction

In recent years the quantity of digitised text has increased substantially, therefore allowing for more complex analysis and processes, including the application of machine learning methods.

Gaining deeper understanding of text has always been a key interest of researchers. One such area is text categorisation. Text categorisation is the automatic assignment of natural language (i.e. human language) to a predetermined categories based upon their content. (Lewis, 1994).

This project classifies a document of English language text into 1 of 20 distinct categories having been trained on the well-known “20 Newsgroups” dataset. This data set contains over 20,000 text documents spread across 20 distinct categories, for example, “motorcycles”, “hockey” and “religion”. These plain-text data take the form of email messages about the specified topic.

To perform this, the data was appropriately pre-processed (including vectorising and tokenising). We then trained the data on multiple models.

use simple models and more advanced neural network models to categorise documents into the relevant categories.

After running the model on the testing dataset found XX to be the best performing model havinf an accuracy of xx, and representing an xxxx performance suggesting the model is able to determine

## Literature Review

## Pre-processing

## Descriptive Analysis

The 20 Newsgroup data set is comprised of approximately 20,000 distributed amongst 20 categories, each with approximately 900 documents in each. The dataset provided had already been split into training (60%) and testing (40%) datasets.A summary of the dataset is shown in Table 1

For effective categorisation the training data should be distributed approximately equally between each group.

Of the training data set there were between 377 and 600 documents with a corpus ranging from 35,869 to 101,427 (with the exclusion of stop-words) for each category. The proportion of the cleaned corpus has a range from 2.6% (rec.autos) to 9.2% (alt.atheism). This is within an order of magnitude so we consider the data to be distributed equally. However, these differences may be a feature we would want to incorporate into our model.

Table

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
|  | **Category** | **Training Number of Documents** | **Training Word Count** | **Average Words per Doc** | **Proportion of Corpus (%)** |  |
| **0** | rec.sport.hockey | 600 | 67385 | 112.3 | 4.9 |  |
| **1** | soc.religion.christian | 599 | 56353 | 94.1 | 4.1 |  |
| **2** | rec.motorcycles | 598 | 50948 | 85.2 | 3.7 |  |
| **3** | rec.sport.baseball | 597 | 48298 | 80.9 | 3.5 |  |
| **4** | sci.crypt | 595 | 42867 | 72.0 | 3.1 |  |
| **5** | sci.med | 594 | 79624 | 134.0 | 5.8 |  |
| **6** | rec.autos | 594 | 35869 | 60.4 | 2.6 |  |
| **7** | sci.space | 593 | 55166 | 93.0 | 4.0 |  |
| **8** | comp.windows.x | 593 | 49193 | 83.0 | 3.6 |  |
| **9** | comp.os.ms-windows.misc | 591 | 54828 | 92.8 | 4.0 |  |
| **10** | sci.electronics | 591 | 72390 | 122.5 | 5.3 |  |
| **11** | comp.sys.ibm.pc.hardware | 590 | 101427 | 171.9 | 7.4 |  |
| **12** | misc.forsale | 585 | 51022 | 87.2 | 3.7 |  |
| **13** | comp.graphics | 584 | 78629 | 134.6 | 5.7 |  |
| **14** | comp.sys.mac.hardware | 578 | 81608 | 141.2 | 5.9 |  |
| **15** | talk.politics.mideast | 564 | 92699 | 164.4 | 6.8 |  |
| **16** | talk.politics.guns | 546 | 85512 | 156.6 | 6.2 |  |
| **17** | alt.atheism | 480 | 125966 | 262.4 | 9.2 |  |
| **18** | talk.politics.misc | 465 | 88406 | 190.1 | 6.4 |  |
| **19** | talk.religion.misc | 377 | 54878 | 145.6 | 4.0 |  |
| **Total** |  | 11314 | 1373068 |  |  |  |
| **Average** |  | 565.7 | 68653.4 |  |  |  |

### Common Words

Following the pre-processing of the text data but before vectorisation we performed several analyses on the data. Once stop-words are removed we found the most common words in the data set.

A picture containing text, clipart

Description automatically generated

As we can see in figure X, the most common words across all categories are generic words with little meaning. These could be candidates to re-classify as stop-words as they are unlikely to differentiate between categories.

By analysing the most frequent occurrences of words in each category we can see if there are real differences between categorisations. Such processes could also identify addition project-specific stop-words which should be removed as part of the pre-processing procedure.

There is a marked difference in the top 10 most common words between each category . For example for the category rec.sport.hockey, the words “*team*”, “*game*” and “*player*” appear most frequently. In contrast the words “*drive*”, “*scsi*”, “*card*” and “*system*” appear most frequently for the comp.sys.ibm.pc.hardware category.

A picture containing text, clipart

Description automatically generated A picture containing text, clipart

Description automatically generated

### Bigrams

Bigrams are pairs of words which appear consecutively in the text. In some cases pairs of words which appear frequently together hold a specific meaning and could have the potential to improve a neural network model.

In this analysis bigrams were identified using *nltk bigram*. The most common bigram “maxaxax..” is a collection of letters found in every document of the *computer* category and therefore could be a significant factor when categorising documents.

Other common bigrams are more familiar combinations such as “dont know”, “law enforcement” and “san francisco”.

Chart

Description automatically generated

Pointwise mutual information (PMI) is a measure of association. **BigramAssocMeasures** from nltk (in conjunction with **BigramCollocationFinder**, a more sophisticated method of identifying bigrams) calculates the PMI, which for the first 20 bigrams gave a value of 20.30. This suggests there are no significant differences between these bigrams.

### Part of Speech (POS) Analysis

In the English language words can be classified according to their grammatical properties. After applying tokenisation we identified the number of nouns, verbs, adverbs, adjectives and other (that is any character that could not be classified into the aforementioned groups) using ***nltk pos\_tag***. For ease of comparability these results were scaled as a proportion of the total number of words in that category.

The POS composition (see figure x) shows each a similar proportion between each of the 20 categories, with the number of nouns accounting for roughly 50% of the total number of words.

A screenshot of a computer

Description automatically generated with medium confidence

## Model Implementation

### Naïve Bayes

Naïve Bayes classifier was employed in this project, which is a kind of supervised learning model. Naïve Bayes is formed by Bayes theorem and naive assumptions of conditional independence between the features. In other words, the Naïve Bayes model structure makes no attribute variable have a significant weight for the decision result, and no attribute variable has a small weight for the decision result. Considering that there are only a few parameters used for model construction, Naïve Bayes is not quite sensitive to missing values in the dataset. In addition, Naïve Bayes is robust even datasets present different characteristics since it has relatively simple logic and model structure. Therefore, Naïve Bayes is widely used for classification tasks, especially text classification. It shows outstanding performance when employed in uncomplicated text classification.

The Naïve Bayes model was constructed through function **GaussianNB()** from **sklearn.naive\_bayes.** The 7580 training records formed by 40 variables were used for training, and a validation set of 3734 rows was used for parameter tuning. The core parameter var\_smoothing is tuned through grid search (please find details in the **Tuning Hyperparameters** part). After seeing the optimal value of var\_smoothing (0.0001), the final model was rebuilt and evaluated by a separate test set of 7537 observations.

### KNN

The second employed model is the k-nearest neighbors algorithm (k-NN), which is a non-parametric supervised learning method. In the KNN algorithm, the classification of a point is determined by majority voting of its neighbors’ classes. And the most common class of the k nearest neighbors determines the class assigned to the point. KNN is one of the simplest classification models. One the parameter of K needs to be tuned.

The KNN model here was built by **KNeighborsClassifier()** function from **sklearn.neighbors**.The same training, validation and test set for Naïve Bayes were also used here. Only K (n neighbors) was tuned with grid search (please find details in the **Tuning Hyperparameters** part). The final KNN model was trained with the optimal K of 1.

### ANN

The Artificial neural network was built here with dense layer 1 of 100 neurons and a ReLU activation function and dense layer 2 of 80 neurons and a ReLU activation function. After that, a Dropout layer was added to prevent overfitting. Another dense layer was constructed with 60 neurons and a ReLU activation function, followed by a dropout layer and one more dense layer (30 neurons). The output layer used the sigmoid activation function. Details can be found in Figure 1 and Figure 2.

Table

Description automatically generated

Figure Sequential model structure.

Diagram

Description automatically generated

Figure Graphical representation of Sequential Model.

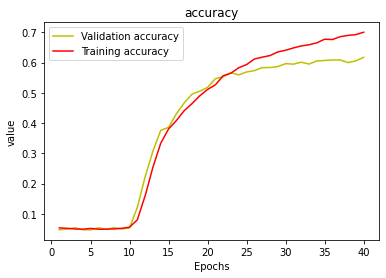


Figure Accuracy and Validation Loss of Sequential Model.

### CNN

The Convolutional neural network is a kind of neural network combined with Convolutional neural network. A convolution layer aims to slide a filter over the input, which consists of a series of learnable filters. As shown in Figure 1 and Figure 2, 1 dimensional CNN was built without using pre-trained embedding. The embedding layer here was constructed by setting input\_dim as 20002 and output\_dim as 200, followed by a Dropout layer with rate of 0.5. Thereafter, two 1d convolutional layers with filters and kernel\_size of 128 and 7, as well as Global Max Pooling operation, were added. We then added a vanilla hidden layer, including a Dense layer and a Dropout layer. As for output layer, we projected onto a multiclass unit output layer, and squash it with a softmax activation function. The model was compiled with multiclass cross-entropy loss function and a rmsprop optimizer. The used package in Python is Keras built on top of Tensorflow.

Table

Description automatically generated

Figure CNN model structure.

Diagram

Description automatically generated

Figure Graphical representation of CNN Model

### Model evaluation

The dataset contains three parts, i.e., training, validation and test sets. The validation set is used to do parameter tuning during the model construction process. And test set is an isolated set to objectively evaluate model performance.

## Tuning Hyperparameters

### Naïve Bayes

The var\_smoothing was tuned with grid search for Naïve Bayes. The var\_smoothing is the portion of the largest variance of all features that is artificially added to variances for calculation stability, which aims to smooths the curve and account for more samples that are further away from the distribution mean. The candidate values for this parameter contain 1e-1,1e-2, 1e-3, 1e-4, 1e-5, 1e-6, 1e-7, 1e-8, 1e-9, 1e-10, 1e-11, 1e-12, 1e-13, 1e-14, 1e-15. The grid search was realized by hold-out validation, which always train model with candidates on training set and test performance with fixed validation set. And according to Table 2, the value of 0.0001 was finally selected.

Table Grid search results for Naïve Bayes.

|  |  |
| --- | --- |
| **var\_smoothing** | **Accuracy** |
| 1.000000e-01 | 0.412962 |
| 1.000000e-02 | 0.414301 |
| 1.000000e-03 | 0.411623 |
| 1.000000e-04 | 0.411355 |
| 1.000000e-05 | 0.411355 |
| 1.000000e-06 | 0.411355 |
| 1.000000e-07 | 0.411355 |
| 1.000000e-08 | 0.411355 |
| 1.000000e-09 | 0.411355 |
| 1.000000e-10 | 0.411355 |
| 1.000000e-11 | 0.411355 |
| 1.000000e-12 | 0.411355 |
| 1.000000e-13 | 0.411355 |
| 1.000000e-14 | 0.411355 |
| 1.000000e-15 | 0.411355 |

### KNN

The K value of KNN represents the count of the nearest neighbors. Suitable K value is important for model training since a small value of k makes model easy to be influenced by while a large value make it computationally expensive. Therefore, the most appropriate K value for this dataset was determined by grid search of hold-out validation. The candidates for K values include integers ranging from 1 to 30. And finally, the value of 1 was chosen here based on the results in Table 3.

Table Grid search results for KNN.

|  |  |  |  |
| --- | --- | --- | --- |
| **K** | **Accuracy** | **K** | **Accuracy** |
| 1 | 0.568827 | 16 | 0.553830 |
| 2 | 0.500536 | 17 | 0.553026 |
| 3 | 0.524103 | 18 | 0.555169 |
| 4 | 0.532137 | 19 | 0.555972 |
| 5 | 0.541778 | 20 | 0.553026 |
| 6 | 0.541510 | 21 | 0.553294 |
| 7 | 0.549813 | 22 | 0.554365 |
| 8 | 0.549813 | 23 | 0.551152 |
| 9 | 0.555437 | 24 | 0.549277 |
| 10 | 0.551955 | 25 | 0.555437 |
| 11 | 0.551419 | 26 | 0.556776 |
| 12 | 0.549277 | 27 | 0.555169 |
| 13 | 0.555437 | 28 | 0.550884 |
| 14 | 0.556240 | 29 | 0.554097 |
| 15 | 0.557847 | 30 | 0.554901 |

### CNN

As for the CNN model, we did not include any complicated grid search work because it is quite time-consuming. The model was trained with epochs of 50 and batch\_size of 64. Early stopping strategy was used that stops model training when validation loss is no longer improving (no better than 1e-2 less for at least 2 epochs). The accuracy and loss for training and validation set during model training is shown in Figure 3.

Chart, line chart, histogram

Description automatically generated

Figure Accuracy and Validation Loss of CNN Model

## Results and Error analysis

### Model accuracy on test set

The model accuracy on the test set is shown in Table 4 below. It is clear that CNN has the best performance, with accuracy of 0.73 while Naïve Bayes has the worst performance with accuracy of 0.39. The accuracy of KNN is slightly higher than that of Naïve Bayes, which is 0.46. And the ANN model has the accuracy of 0.58.

Table Model accuracy on test set.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Model** | **Naïve Bayes** | **KNN** | **NN** | **CNN** |
| **Accuracy** | 0.39 | 0.46 | 0.58 | 0.73 |

A description of the key evaluation metrics are as follows.

Precision is

Recall is

The F-score (or F1-score) is a measurement of the accuracy of a binary classifier built upon the precision and recall metrics already mentioned and is defined as follows;

whereby TP is the number of true positives classified, FP is the number of false positives and FN is the number of false negatives classified by the models.

### Confusion matrix

Effectiveness by group for best performing model

|  |  |  |
| --- | --- | --- |
|  |  |  |
|  |  |  |
|  |  |  |

## Conclusion

## References