News Articles Classification

COMP9417 Machine Learning and Data Mining

Group Name: 91-DIVOC

Group Members:

1. Kanadech Jirapongtanavech (z5176970)
2. Harris Trong Nam Phan (z3415157)
3. Jaydon Tse (z5214494)
4. Saksham Yadav (z5164624)
5. Farhan Ghazi (z5199861)

Introduction

Explanation

We live in a world where people read news articles all the time. Certain users have preferences when it comes to reading articles. Some people are more interested in reading sports articles while others might like to read about health. Hence, we want to recommend articles to users and one way of doing that is categorising articles into topics. In turn, recommending articles with a particular topic means that the user will be more interested in the article hence they will click on it. More clicks means more profit for the publisher.

Aim

To suggest up to 10 of the most relevant articles from the test set of 500 articles to users through the use of machine learning models.

Issues that will need to be addressed:

Managing the Irrelevant articles

- Irrelevant articles do not belong to any of the topics hence will need to be excluded when it comes to recommendations.
- In our case, we will still use 'Irrelevant' as a target variable. So it will be used in the training and test sets for our algorithms.
- However, 'Irrelevant' will not be included in our final report.

Having more than 2 classes that need to be considered

- Having 10 topics as the target variable means that some machine learning models will be considered.
- This means models such as logistic regression and binary naive bayes will not be used. Regression models will not be an option as well.
- In our analysis, we instead opt for Neural network based approaches, whether it will be using Keras or Scikit learn Neural Network libraries.

Manipulating the given training and test datasets

- In the training and test sets that were given to us, the column 'article_words' is a list of words that were separated by commas.

- Even though all the stop words were eliminated, the column still required some manipulating.
- The words needed to be split and then for each reference we counted how many times a particular word showed up. Each word becomes a variable, and there were quite a lot of them.
- For our neural network model, we converted words to an n-gram vector representation (referred to as bag-of-words approach). Doing so, enabled the network to discard the ordering of words within text as focus merely on classification.

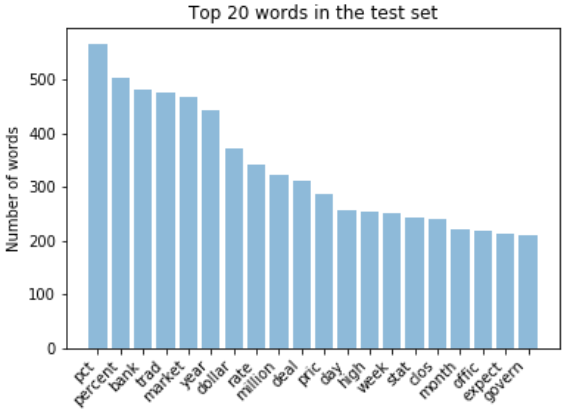
Computational speed

- The sheer amount of columns that came as a result of the number of unique words were an issue for computational speeds. This means trying to filter out words which do not meet a particular condition.
- This includes getting the first 2000 or 20000 words with the highest frequency in all of the training set.
- Another method could be filtering for words that exceed 5 characters in length.

Exploratory data analysis

Most frequent words

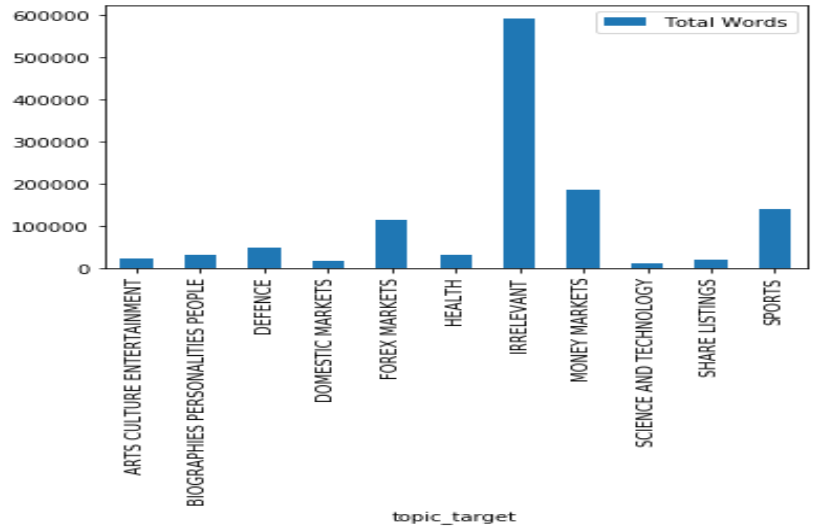
In our preprocessing we want to see which words come up the most regardless of the class. Here is a bar chart of the top 20 words in the training set:



So the Top 20 words for the training set and test set when it comes to distribution are in fact quite similar to each other. This bodes well when we want to train the model and then test it out as this means less chance of overfitting on newly seen data.

Class distribution

We will take the class distribution of the training set and test set separately. So the bar graphs look like this. We look at the total number of words per topic in the training set.



As you can see, most of the words overall come in the irrelevant section and this might pose problems when we are trying to categorise articles. This comes as a result of Hence this will need to be managed. However, we will still continue to use the 'Irrelevant' class in our modelling.

Summary statistics

We look at the summary statistics for each of the 10 classes. Here is what it looks like on a topic level for the training set

topic_target	count	mean	std
ARTS CULTURE ENTERTAINMENT	117.0	178.803419	108.835231
BIOGRAPHIES PERSONALITIES PEOPLE	167.0	185.311377	101.385814
DEFENCE	258.0	186.344961	89.174125
DOMESTIC MARKETS	133.0	129.345865	80.836540
FOREX MARKETS	845.0	136.029586	104.188322
HEALTH	183.0	162.688525	82.810363
IRRELEVANT	4734.0	125.294043	98.816131
MONEY MARKETS	1673.0	110.899582	92.838296
SCIENCE AND TECHNOLOGY	70.0	164.242857	77.396483
SHARE LISTINGS	218.0	95.389908	79.247784
SPORTS	1102.0	127.389292	99.886275

So in addition to the data analyst above, irrelevant makes up most of the article classifications in the training set. On the other hand, there are not a lot of observations that are related to 'Science and Technology' and 'Arts Culture Entertainment', which means there might be some difficulty finding a classification for the test set. Having more words per article on average will help as well when it comes to classifications.

Methodology

Model A: Support Vector Machine Model (SVM)

Generalised Model / Method Overview

Support Vector Machine, also known as **SVM** is a **supervised** machine learning algorithm. This method works by plotting data items in **n-dimension space** with each item being a particular **coordinate (point)** on this space. Afterwards, we find an (n-1) dimensional **hyper-plane** that best separates the data points into their respective categories.

Method Selection

1. **Memory efficient** - Since we do not own a computer specialised for data analytics, this method can reduce the time and load on our computers as we process the data.
2. **High dimensions accuracy** - The data that was given to us has it that each different word is its own feature, making the process to require high dimensional spaces. Thus, it would be wise to use a method that works well with high dimensions
3. **Training speed** - since the classifier is linear, its training speed is much faster than ensemble models like boosting tree classifiers while having similar accuracies.

Feature Selection

As a first step in preparing the data for training, we used the TfidfVectorizer module to transform words to feature vectors that can be used as input to the SVM model. TF-IDF (Term Frequency-Inverse Document Frequency) is a highly popular algorithm to convert textual data into a numeric form which can be understood by machine learning models. It is a numerical statistic that depicts the importance of a particular word in a document, or collection of documents.

Hyperparameter Tuning & Selection

After processing the data using TF-IDF vectorization, we ended up with a lot of words that are useless in effectively classifying article topics. As a result, we experimented with various thresholds for the ‘max_features’ parameter (the maximum number of words to use for classification). It turns out that using a maximum of ~2000 features (words) results in the highest performing model, with an accuracy of 77%.

Model B: Multi-Layer Perceptron (MLP)

Generalised Model / Method Overview

Multilayer perceptrons are a class of feedforward Artificial Neural Network **(ANN)**. This model works by having layers of neurons: typically an **input** layer, some **hidden** layers and a final **output** layer. The input layer is the first layer, the one that takes in data. The input layer sends the data to the hidden layer(s) where it is processed in some manner before the network makes a **classification** prediction for the data point. The predicted output is compared to the expected output. If they are the same, the **'weights'** of the neurons in the hidden layer remain the same. However, if they differ, the 'weights' within the hidden layers would be altered (by the process of **backpropagation**) so as to push the model towards making a better, more accurate prediction. The model repeatedly does this, improving its performance over epochs, until it is able to correctly classify future unknown data.

Method selection

We'd like to acknowledge that much of our incentive behind adopting an MLP model comes from **Google's Machine Learning Developer Guide** (Introduction | ML Universal Guides | Google Developers, 2020)**.** The Google ML Guide is the culmination of key insights derived from running a large number (~450K) of machine learning experiments across problems of different types (especially sentiment analysis and topic classification problems), using 12 datasets, alternating for each dataset between different data preprocessing techniques and different model architectures. A core metric we used to choose our method was the **number of samples/number of words per sample ratio.** We calculated this metric to be less than **1500** for the given dataset and as per the guide, decided to proceed with an MLP model for the given text classification task. *In general, research suggests that MLP's have superior classification performance on datasets of similar size to the one given to us.*

Feature Extraction

Feature extraction mainly involved effectively converting the corpus to a **strict numeric representation**, since machine learning models take numbers as inputs. The stock dataset provided to us had already been **lemmatised.** In the case of our MLP model, it was vital for us to ensure the following: *the order of words within the text should in no way influence model learning*, since, after all, we are trying to build a *classifier.* To achieve this, we adopted a '**bag-of-words'** approach converting each word to a **token,** which was then further grouped together into a **(uni-gram, bi-gram) vector representation.** The above

vector representation groups together adjacent tokens within the text into unique n-grams (in our case, into unigrams and bi-grams) which we then converted to **numeric vectors** that our model can work with. The real benefit of this preprocessing method lies in the fact that doing so effectively enables the model to **completely disregard the order of words in the text** (which is what we desire).

Feature Selection

After data preprocessing (as discussed above)**,** we ended up producing tens of thousands of **tokens.** However, as is the case with all machine learning methods, it is important to understand that not all data that is available translates to information. Hence, for our feature selection step we used **ANOVA F-value** as a baseline to pick the best subset of tokens to train our model on. *The ANOVA F-value is a standard statistical scoring method used to rank how much a token contributes to label prediction*. Using this scoring method, we picked the top **20,000** highest-scoring tokens as our training features for the neural network.

Hyperparameter Tuning & Selection

We trained a 2-layer perceptron model with the following build characteristics:

1. **Input Layer**: 'Dense' layer (32 units)
2. **Hidden Layers**: As many drop-out layers as 'Dense' layers each with 32 output dimensions (drop out layers perform regularization at rate of 0.2; Dense layers use **'ReLU'** as activation function (computationally less costly)
3. **Output Layer**: 'Dense' layer with 11 output 'layer units' (since we have 11 topic articles to predict)
4. **Activation Function**: **Softmax** (standard for text classification tasks)
5. **Loss Function**: **Sparse Categorical Cross Entropy** (standard for text classification tasks)
6. **Optimizer**: **Adam** (latest proven optimizer)

The above model hyperparameters are the **optimal** parameters which were deduced after much research and experimentation.

Evaluation Metrics

Besides training set validation accuracy and overall test set accuracy, the core evaluation metrics we calculated and their significance are summarised below. In simple terms, the metrics are defined as follows:

- **Precision Score:** It is the number of relevant objects classified correctly divided by the total number of relevant objects classified.

  Precision = TP / (TP + FP), where TP = True Positives, FP = False Positives
- **Recall Score:** It is the number of relevant objects classified correctly divided by the total number of relevant objects.

  Recall = TP / (TP + FN), where FN = False Negatives
- **F1 Score:** It is a measure of accuracy, which is simply the harmonic mean of precision and recall scores, that is,

  F1 = 2 * ((precision * recall) / (precision + recall))

The above metrics for the actual **test set** have been summarised in the '**Results'** section of the report. Please refer to the 'Results' section for an in-depth look into the metrics obtained for **BOTH** models.

Results

Cross-validation results for SVM model

Validation on **10%** of train set yielded following results:

Validation Acc: 0.7638495068732541,

Loss: 0.57498455631814

Final results for SVM model

Results on the training set for selected metrics, feature sets and implemented methods.

Topic Name	**Precision**	**Recall**	**F1**
ARTS CULTURE	0.40	0.67	0.50

ENTERTAINMENT			
BIOGRAPHIES PERSONALITIES PEOPLE	1.00	0.20	0.33
DEFENCE	0.80	0.62	0.70
DOMESTIC MARKETS	N/A	N/A	N/A
FOREX MARKETS	0.42	0.29	0.35
HEALTH	0.75	0.86	0.80
MONEY MARKETS	0.50	0.65	0.57
SCIENCE AND TECHNOLOGY	N/A	N/A	N/A
SHARE LISTINGS	0.75	0.43	0.55
SPORTS	0.95	1.00	0.98

Cross-validation results for Multi-Layer Perceptron Model

Validation on **10%** of train set yielded following results:

Validation Acc: 0.7736842036247253,

Loss: 0.56085332167776

Final results for Multi-Layer Perceptron Model

After **training**, model performance on test set is as follows:

Accuracy: 0.7839999794960022

Loss score: 0.5499200468063354

Here are the **key metric evaluations** over the whole **test set:**

Topic name	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	0.33	0.67	0.44
BIOGRAPHIES PERSONALITIES PEOPLE	0.60	0.20	0.30
DEFENCE	0.89	0.62	0.73
DOMESTIC MARKETS	0.50	0.50	0.50
FOREX MARKETS	0.50	0.35	0.41
HEALTH	0.90	0.64	0.75
MONEY MARKETS	0.49	0.67	0.57
SCIENCE AND TECHNOLOGY	0.00	0.00	0.00
SHARE LISTINGS	0.50	0.43	0.46
SPORTS	0.95	0.98	0.97
ACCURACY			0.79
MACRO AVERAGE	0.59	0.54	0.55
WEIGHTED AVERAGE	0.77	0.77	0.76

Final article recommendations

We have decided to use the ***Multi-Layer Perceptron Model*** as our final classifier. The final article recommendations and their metrics are shown below.

Topic Name	Suggested Articles	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	9703, 9952, 9789	0.33	0.50	0.40

BIOGRAPHIES PERSONALITIES PEOPLE	9896, 9940,9988, 9854, 9526,9604, 9933,	0.26	0.33	0.29
DEFENCE	9576, 9616, 9559, 9773, 9670, 9842, 9770	0.45	0.50	0.47
DOMESTIC MARKETS	9994, 9640	0.25	0.50	0.33
FOREX MARKETS	9693,9632, 9875, 9729, 9584,9748, 9786, 9704,9986, 9551	0.18	0.33	0.24
HEALTH	9810, 9661,9929, 9735,9621, 9911, 9873, 9947,9617, 9982	0.44	0.50	0.47
MONEY MARKETS	9765, 9871,9769, 9618, 9755,9707, 9602, 9516,9761, 9638,	0.27	0.52	0.36
SCIENCE AND TECHNOLOGY	NIL	0	0	0
SHARE LISTINGS	9518, 9601,9666, 9972, 9715	0.44	0.67	0.53
SPORTS	9942, 9596,9848, 9857, 9568,9752, 9760, 9774,9569, 9813	0.91	0.95	0.93

Discussion

Compare different methods, their features and their performance. State any general observations. Discuss the metrics in the above two tables. Which metric(s) is/are more appropriate and why?

Model A (Support Vector Machine) is a supervised machine learning algorithm while Model B (Multilayer Perceptron) is an Artificial Neural Network.

From the key metric evaluations, the performance of the two models did not differ too much in terms of recall. For precision, model B is slightly higher on all topics buy share listing, and biographies personalities & people than model A. For F1, model A is slightly higher on all topics except for defence and forex markets. Overall, model B has a better accuracy than model A.

The importance of each metric is highly dependent on the business scenario. A high accuracy can be obtained by simply assigning all observations with the most frequently occuring class, however, this would be pointless as there would be no differentiation between any classes. The precision metric attempts to answer the question 'What proportion of positive identifications was actually correct?' Similarly, recall attempts to answer ‘What proportion of actual positives was identified correctly?’ The F1 Score is simply a harmonic mean of the precision and recall. In this particular scenario, we would argue that precision and recall are more important than accuracy, as they penalize incorrect classifications to a greater extent. For example, it is more important that readers who are interested in finance, receive correct recommendations for finance, instead of incorrect sports recommendations, which may give a higher accuracy as there are more instances of sports articles.

What can be done to improve method, features, performance?

For Model A (SVM), performance could be enhanced by improving preprocessing methods, such as grouping common words (e.g. like, liking, liked), which will allow the model to more effectively classify article topics. Furthermore, if we had more computing resources available, we could try different SVM kernels such as polynomials (of varying degrees).

In the case of Model B (MLP), performance could potentially see a significant boost if we had access to a much larger training set. Neural networks tend to see performance improvements with more data to train on. Since the model trains on features which are sourced from words within the text, a larger corpus of text may produce more insightful features for the neural network to train on, leading to more accurate predictions.

Which methods are potentially useful but haven’t been tried?

In the field of text classification, there are a multitude of different models that could have been tried out such as Multinomial Naive Bayes (MNB) and LSTM Recurrent Neural

Networks (deep learning). Another potentially useful method is stacked generalization. The stacked generalization is an ensemble model whereby cross-validated predictions of multiple base estimators are stacked to form the first layer of the method. The final estimator will train using stacked predictions of the first layer and output the final prediction. This method is believed to utilise the strength of each base model in the first layer.

Which parameters are potentially useful but haven't been tried?

Looking at the frequency bar chart in the exploratory data analysis section, it seems like there are words which are basically the same as each other, hence they should be grouped together. For example, pct and percent are the same thing, so that is one thing that could be grouped together into 1 variable instead of 2. This allows us to clean up the dataset once more, and with new, cleaned up variables it might improve the prediction of the model.

Another way to do it is to group technical words together. So if words come under a specific topic, that can be grouped together as well. So for example, if we talk about dollar and cent (which showed up in the frequency bar chart above), that is likely to refer to the markets. Grouping it in such a way makes the grouped variable more topic focused, and hence the model might do better when it comes to making predictions on the topic.

However, this will require us to look through every unique word in the training and test set, which can take up quite a lot of time.

Conclusion

In this **Natural language Processing (NLP) classification project**, we were tasked with implementing a topic classifier that is capable of correctly classifying different articles belonging to 11 unique topics. This is an excellent example of the use and application of machine learning methods in the real world. The ideas and methodology we developed here can be further extended to a much wider problem space, such as that of sentiment analysis as well as speech recognition. Some of the more technical aspects that we can draw conclusions from are discussed below:

To train an NLP model it is important to perform preprocessing such as stop words removal, stemming and lemmatization to engineer useful features. In addition to these basic preprocessing techniques, other techniques like bag-of-words and tokenization were also applied. Oversampling of minority classes was also applied to combat class imbalance issues, but the results were not significantly different as such it was omitted from the report.

Other machine learning models such as XGBoost and Random Forest were also used to classify article topics, but they did not yield better accuracy than basic models like linear SVC (model A) or basic neural network (model B). As such, we have learned that features engineering are extremely important in any classification task and that extra effort should be put in data exploratory and features extraction in the future.

Reference

Google Developers. 2020. Introduction | ML Universal Guides | Google Developers. [online] Available at: <https://developers.google.com/machine-learning/guides/text-classification> [Accessed 27 April 2020].