News Article Recommendation Matrix

A Problem in Text Classification

COMP9417 Project by 94 Smash

Kai-Yeh Yang (z5125244), Clare Xinyu Xu (z5175081), Sarah Williams (z5205420), Hayden Sip (z5118630), Andrew Cvetko (z3330071)

Introduction

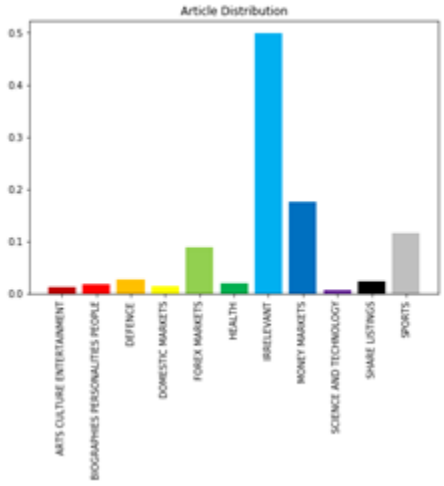
The volume of news content has grown at a significant pace over the last decade. The sheer quantity of articles makes it humanly impossible to find and read the most relevant articles on a certain topic. As such, recommendation systems have been developed to choose articles amongst the set of all articles that are most relevant in a certain topic or that a user would find interesting. In this paper we deal with a specific problem in this field. Given ten users, recommend a list of at most ten articles that matches each user's specific topic of interest. The aim of this paper is to achieve a solution to this problem with a high degree of accuracy using algorithms within the classification field of machine learning.

We considered three main tasks that required our attention in tackling this problem. They were the feature engineering, parameter tuning and overfitting prevention for each model. Descriptive statistics were extracted from the distribution of words within the set of training articles to construct vectorised features through a *bag of words* and *term frequency - inverse domain frequency* (TF-IDF) representation. These vectorisations were then fed as inputs into the various machine learning models considered whilst exploring solutions to the problem space. We tailored searches over the parameter space for each model to optimally determine the best combination of hyperparameters during each model training. Particular care was taken to ensure model performance was consistent across the training and testing sets. This ensured each model was not only accurate, but robust to noise and free from overfitting. This process was repeated with the aim of improving the final model's article recommendation accuracy.

The paper is broken down as follows; exploratory data analysis will look at the underlying training data to develop insights into the development of features to be used in the models. This will be followed by a discussion focused on the choices made for each selected machine learning model, a presentation of the preprocessing of data and features used, tuning of hyper-parameters and an evaluation of model performance. Final results on unseen test data will be presented along with an outline of the model parameters used. This will be followed by a discussion and comparison of the results and models and concluding remarks.

Exploratory Data Analysis

The training data consisted of 9,500 samples, with three columns: an article number, preformatted article contents in the form of comma separated strings and a topic class label. A distribution of the articles shows that IRRELEVANT topics account for close to 50% of all articles in the set. This is followed by MONEY MARKETS, SPORTS and FOREX MARKETS with 17.6%, 11.6% and 8.9% respectively. All other topics have less than 3% representation in the training set (see Appendix 1). Given the disproportionate distribution of articles, there may arise issues of the training models to be biased towards the more frequent article topics and not learn the nuances of the lower represented articles. This behaviour would reduce our prediction model accuracy if the test set followed a different class distribution. A solution to this generalised test set would be to over sample the smaller classes and under sample the larger classes whilst training. The brief confirms that the test and training sets follow similar distributions, hence normalising set size inconsistency was not deemed necessary.



Feature extraction was made simpler as the training set input text was already cleaned and stemmed to the root word. To further generate features two preprocessing methods under the sklearn library were considered. CountVectorizer converts a text document into a sparse matrix noting the frequency of words in each article in the corpus. A consequence of the vectoriation is that word ordering is not preserved, effectively creating a bag of words. Examining the data further revealed that some of this information appeared to be lost given the already cleaned training data.

Under the bag of words approach (using default parameters) we can start to pull in some insightful information to help in model building. There are 1,213,214 words in the training corpus with 35,822 of these being unique. Each of these words represents a distinct feature that could be used. Given that there are only 9,500 training samples, caution was needed to prevent overfitting this many features to our models. Looking at the most frequent words across each topic reveals a natural tendency for these to be closely associated with that topic. For example, under the topic of DEFENCE we have nato, stat(e), milit(ary), forc(e), and minist(er) being the top 5 most common words all of which all have connotations to defence. Appendix 2 outlines the 10 most common words and their frequencies.

A concern from this analysis is the commonality in the most frequent words across some topics as more frequent words are likely to have a larger impact in the classification to a particular topic For example majority of the top 10 most frequent words are the same across FOREX MARKETS, IRRELEVANT and MONEY MARKETS articles such as trad and market. This poses a challenge as how to correctly classify the article if the words across different topics have similar composition and distribution.

FOREX MARKETS		IRRELEVANT		MONEY MARKETS	
dollar	3,358 (2.92%)	percent	5,945 (1.00%)	bank	4,447 (2.40%)
bank	2,178 (1.89%)	year	5,237 (0.88%)	rate	3,772 (2.03%)
rate	1,755 (1.53%)	million	4,740 (0.80%)	percent	3,515 (1.89%)
trad	1,678 (1.46%)	trad	3,678 (0.62%)	dollar	3,387 (1.83%)
market	1,668 (1.45%)	market	3,556 (0.60%)	market	2,901 (1.56%)
percent	1,336 (1.16%)	shar	3,294 (0.56%)	trad	2,490 (1.34%)
currenc	1,292 (1.12%)	compan	3,190 (0.54%)	day	1,775 (0.96%)
deal	1,208 (1.05%)	stat	3,160 (0.53%)	deal	1,707 (0.92%)

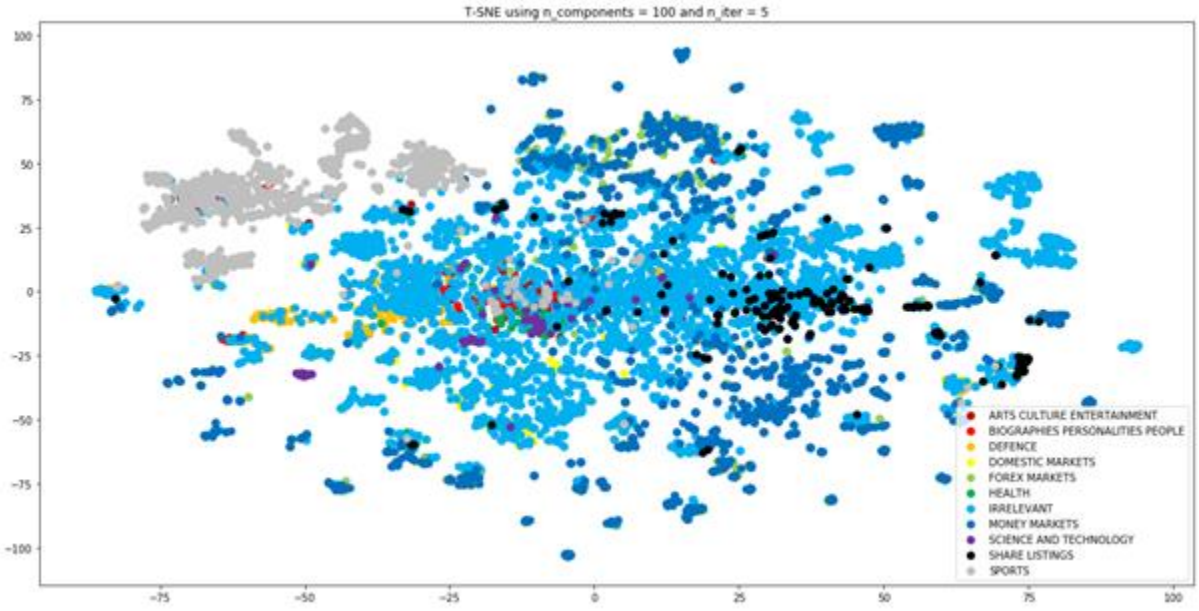
mark	1,191 (1.04%)	bank	3,095 (0.52%)	cent	1,533 (0.83%)
yen	962 (0.84%)	pric	2,611 (0.44%)	currenc	1,472 (0.79%)

A potential avenue to solve this problem is the second preprocessing method considered being TfidfVectorizer (term frequency inverse document frequency TF-IDF). TF-IDF weights each word in a document based on its importance or how frequently it appears but offsets this against how frequently this word appears in other documents in the entire corpus. As such, it focuses on words that appear frequently, yet remain unique to that topic. Undertaking the same frequency analysis showing the words with the highest TF-IDF in each topic reveals a greater dispersion of words whilst maintaining the relevance of words in each topic. This indicates that TF-IDF may be a better choice in processing features compared to bag of words. A summary of the highest rated words in each topic under TF-IDF is provided in Appendix 3.

So far we have considered only the content of the words in the articles. Context is also important as a sequence of words can provide more information to help in classification. CountVectorizer and TfidfVectorizer both support n-grams which is an option to look at all sequences of words. Looking at the frequency analysis under n-grams revealed only some sequence of words to be amongst the highest frequency/TF-IDF scores in which case they were the same word doubled up. Upon further inspection in the training data this was indeed the case with the same words grouped together and we concluded that the context of article words had been lost in the initial preprocessing of the training data.

Given the large amount of features generated, reduction of the feature space was considered to help with potential overfitting within the models. Again CountVectorizer and TfidfVectorizer both have options available to reduce the feature space to ensure only the most important words are considered. These include min_df (ignores words with a document frequency below this threshold), max_df (ignore words with a document frequency greater than this threshold) and max_features (only include up to this many features) which were considered in the training of the models. Another approach considered was the use of dimensionality reduction using the t-SNE library in sklearn. t-distributed stochastic neighbour embedding is a visualisation technique that applies a nonlinear dimensionality reduction technique (truncated SVD) to embed high dimensionality data into a lower dimension.

The visualisation below shows the distribution of classes in two dimensions after truncated SVD has been applied with n_components = 100 and n_iter = 5. Other combinations of these inputs were run but the output was broadly the same. A few things standout. One; SPORTS articles seem to be clustered together. This may lead to better performance in the models in classifying this class given the homogeneity in this class. IRRELEVANT articles are widely dispersed and dominate the chart. The consequence of this in modelling is that the models may predict more IRRELEVANT articles as it cannot segregate out this from other classes. Finally, there doesn't appear to be any clear clustering of other classes again suggesting potential difficulty in classifying articles correctly.



Methodology

To tackle the problem of text classification within a recommendation system, we tested multiple classification models to figure out which one best fit the data. The models tested included multinomial logistic regression, multinomial naive bayes and support vector machines. Features and hyper-parameters were tuned on each model individually, and the method for each model is described below. Our chosen sklearn models require an numerical array-like input to fit and classify the data into classes. Therefore all our models required a vectorised count mechanism provided by a 'bag of words' representation of the input data. This representation transforms the input such that all unique features of the model are represented by a series of bytes, which form a vector. The bag of words results in a compact sparse array of features which is readily accepted by our estimators.

Multinomial Logistic Regression (MLR)

Multinomial logistic regression is a well-used text classification model, based on classifying an instance to the class with which it has the highest probability of belonging to, as defined by the equation trained via the model. We chose this model because of its ability to consider the weights of individual features. To create our model, we considered two of sklearn's logistic regression classifiers: LogisticRegression and LogisticRegressionCV. We transformed our dataset into a bag of words. A consequence of this vectorisation process is that word ordering is not preserved, however a brief examination of the training set reveals that this information had already been lost because of its chosen preprocessed format. For each feature we generated the Term Frequency - Inverse Document Frequency (TF-IDF), which indicates the relative importance of a word in the dataset. This was implemented by sklearn's TfidfVectorizer. The benefit of using TF-IDF in feature engineering is that it is a logical metric to extract the most descriptive terms in a document.

To further improve the efficiency of our model, we performed dimensionality reduction through Truncated -Singular Value Decomposition (T-SVD). This is a matrix decomposition method that breaks down the matrix into singular values, and selects the features based on the sigma value (variance). The key advantage of dimensionality reduction, as implemented via T-SVD, is that it reduces computational time, and improves visualisation of results. When T-SVD is applied to the feature space, the search finds that 95% of all distinguishable deviations are contained in a reduced dimensionality of 2708 components from 7000 and greater, TF-IDF features. This information was supplied with a variance of 0.95, when removing words from less than 10 papers.

We also considered n-gram tuning when generating TF-IDF scores. N-grams convert the singular word features into a sequence of words, n features long. In our feature engineering process, it was found that this made no difference to the accuracy of our model. This is a direct consequence of the earlier conclusion that word order has not been preserved in the preprocessed input data. In turn, we decided to disregard this method, to reduce unnecessary complexity within the model, guided by Occam's Razor principle.

Another method that we investigated but did not implement for MLR, is the use of ensemble methods, particularly, boosting. This is because in logistic regression the algorithm will always generate a model, with the lowest possible residual. Boosting then, does not provide any additional benefit as the minimum

residual was already found. Hence, to again eliminate unnecessary complexity, this method was not implemented.

The MLR model's hyperparameters were tuned to ensure the most influential parameters were tuned, with an aim to reduce overfitting on the training data. The key hyperparameters tuned were the cross-validation size and regularization size. Here, we considered two cross validation classes to perform the hyperparameter tuning of our model, LogisticRegressionCV and GridSearch. LogisticRegressionCV was chosen as it is optimised to perform a warm start search across a range of hyperparameter 'C' values for this linear model. It is much quicker and produces much more reliable results then GridSearch. To optimise our final model we tuned across 100 logarithmically spaced C values with an exponent dynamic range of ±20 (e-20 to e+20). The search was performed against the four different solvers capable of handling multiclass regressions - 'newton-cg', 'lbfgs', 'sag' and 'saga'. The best model used the lbfgs solver, with 2000 maximum iterations producing a training result accuracy of 0.857 and a test result accuracy of 0.774. This was eventually determined to be our best performing model.

We also considered the effect of performing two different kinds of bagging on our standalone LogisticRegressionCV model. One with all features present and the second with random samples of features without replacement. Both models were bagged with samples stochastically picked with replacement. The model performed equally as well as the 'lbfgs' solver variant, and hence was discarded as unnecessary model complexity.

Multinomial Naive Bayes

The premise behind the Naïve Bayes classifier, in the field of text classification, is to find the probabilities of classes assigned to texts by using the joint probabilities of words and classes. There are three naïve bayes classifiers implemented in sklearn: GaussianNB, BernoulliNB, and MultinomialNB. We choose the MultinomialNB model whose concept is to count how often a word appears in a sentence due to the speed of training and prediction as well as its performance with high-dimensional data.

As with the MLR model we undertook feature engineering, obtaining features represented by both TF-IDF and bag of words count vectors. We observed the model performed better with features obtained under the bag of words, which we believed to be due to posterior probabilities of the classes being more accurately represented conditional on the direct bag of words count vectors. We proceeded to train the models with the bag of words count vectors.

To prevent overfitting, we explored hyperparameter optimization. The alpha smoothing parameter was considered for this problem as there is no specific prior data other than the training set and the prior distributions of the classes are believed to follow multinomial distribution. Here we choose two methods of tuning. Firstly, we performed a random search with sklearn RandomizedSearchCV which is to search the hyperparameters randomly to decrease the processing time. The second way is grid search. Instead of generating the random values, it works by searching the specific hyperparameters. The advantage of a grid search is that it can find the optimal parameter combination however, it's very time-consuming. We determine optimal value for alpha through the above methods, evaluating 3-fold cross validation with 50 iterations to evaluate the performance of the model at each value. We obtained the optimal value of alpha to be 1.44, with a training set accuracy of 0.817 and a test set accuracy of 0.72.

Furthermore we also tested bagging which can lower the variance to improve the accuracy of the model. It achieves this by creating random subsets drawn from the training set and aggregating the predictions made on the model across each subset. Sklearn's implementation of bagging ensemble is BaggingClassifier, whose primary running hyper-parameter is the number of base classifiers created.

Support Vector Machines(SVM)

Support Vector Machines is a supervised machine learning algorithm that works by selecting the data points and defining the boundaries between them. We considered using this technique as the SVM algorithm is suitable for both linearly and nonlinearly separable data.

We tuned the SVM by changing the hyperparameters C, gamma and the kernel function. C represents the bearable error of the misclassification error whilst gamma evaluates the distance of the calculation of the separation line.We have tested a wide roughly range of C and gamma, while the range of C is from 0.001 to 0.01 and the latter, gamma, between 0.0001 and 100. Lastly considered parameter is kernels, which takes low dimensional input and transforms into the high-dimensional one. Kernel functions considered here were 'RBF', 'Sigmoid' and 'Linear' respectively. Sklearn's GridSearchCV function was used to tune the SVM hyperparameters as it assists to build a combination algorithm parameter in the grid.According to the result, the best_svc model is with C at 0.0005, degree at 3, gamma at 0.005 with the kernel function 'rbf'.

In general, we perform the 3-Fold Cross Validation on both SVM and Multinomial NB method search. The reason is that we would like to find the balance between the execution time and the number of models that have been testing.

Evaluation Matrix

The idea of evaluation metrics is to explain the performance of the model. The progress will be made according to the feedback from metrics until a desirable accuracy is achieved. The choice of metric completely depends on the type of models. We perform several of the metrics on the classification template which works well on the binary classifier. They show as following:

Accuracy: It defines as the percentage of queries in which the engine is able to predict the correct label, which is one of the simplest forms of evaluation metrics.

Precision: Precision is a metric for binary classifiers which measures the correctness among all positive labels.

Recall: Recall measures how many positive labels are successfully predicted amongst all positive labels.

F1-score: F1-Score is the harmonic mean of precision and recall values for a classification problem. It is followed the formula by F1 = 2 * Precision * Recall / (Precision + Recall)

However, for the problem where there are multiple values, we first have to transform our problem into a binary classification problem. Besides, since accuracy is a simple way to measure the model, we first calculate it to detect whether our model is overfitting or not both on training and testing data. We have also obtained the confusion matrix from sklearn.metrics. A confusion matrix is the summary of prediction results which gives insight not only into the errors being made but more importantly the types of errors.

Results

Results of Multinomial Logistic Regression Model on test set

Topic	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	33.33%	33.33%	33.33%
BIOGRAPHIES PERSONALITIES PEOPLE	20.00%	100.00%	33.33%
DEFENCE	61.54%	100.00%	76.19%
DOMESTIC MARKETS	50.00%	100.00%	66.67%
FOREX MARKETS	31.25%	48.39%	37.97%
HEALTH	42.86%	75.00%	54.55%
MONEY MARKETS	66.67%	53.49%	59.35%
SCIENCE AND TECHNOLOGY	0.00%	0.00%	NAN
SHARE LISTINGS	28.57%	50.00%	36.36%
SPORTS	96.67%	95.08%	95.87%

Article recommendations using Multinomial Logistic Regression

Topic Name	Suggested Articles	Precision	Recall	F1
ARTS CULTURE ENTERTAINMENT	9703 9933	100.00%	33.33%	50.00%
BIOGRAPHIES PERSONALITIES PEOPLE	9940 9526	50.00%%	100.00%	66.67%
DEFENCE	9559 9576 9770 9616 9773 9987 9842 9670	100.00%	100.00%	100.00%
DOMESTIC MARKETS	9994	100.00	100.00%	100.00%

FOREX MARKETS	9986 9875 9977 9529 9718 9530 9682 9588 9748 9772	100.00%	50.00%	66.67%
HEALTH	9661 9873 9929 9810 9609 9937 9735 9621 9982	100.00%	66.67%	80.00%
MONEY MARKETS	9618 9871 9765 9755 9769 9761 9998 9602 9516 9835	58.33%	70.00%	63.64%
SCIENCE AND TECHNOLOGY		0.00%	0.00%	NAN
SHARE LISTINGS	9518 9601 9666 9972 9715	100.00%	50.00%	66.67%
SPORTS	9857 9886 9573 9620 9569 9752	100.00%	100.00%	100.00%

	9760 9848 9787 9596			

Results 8 marks – presentation of results using figures and tables

Result evaluation using appropriate metrics

Evaluation of various hyper-parameters and design choices

Analysis of feature importance

Cross validation results on the training set for selected metrics, feature sets and implemented methods

Final results for each class calculated on the whole test set using the final selected method with its hyper-parameters

Final article recommendations using the final selected method with its hyper-parameters. In addition, please calculate precision, recall and F1 for these recommendations

Discussion

Best model selection

Model	Training Set Accuracy	Testing Set Accuracy	F1 score
Multinomial Logistic Regression (with and without bagging)	86.1%	77.4%	75%

Multinomial Naïve Bayes	81.7%	72.0%	71.0%
Multinomial Naïve Bayes with Bagging	81.0%	72.4%	71.0%
SVM	49.8%	53.2%	51.6%

In summary, MLR and MNB perform well regarding the training and testing set accuracy. It is also observed that SVM seems to underfit since the accuracy of the testing set is higher than the training set after model training. Upon closer inspection, we think that SVM performs poorly and takes a long time to execute because of the large size of our dataset, as this type of model is best suited for smaller datasets (E. L. Iglesias,R. Romero,and L. Borrajo 2015). Furthermore, the accuracy is too low compared to the other models so we will discard it. We have also conducted the ensemble method bagging on MLR and MNB. There is no difference in the accuracy of the training and testing set for logistic regression whether bagging has been applied or not. As for MNB, there is a slight improvement in the testing accuracy by 0.4% while the training one decreases by 0.7%. The main reason behind the interesting accuracy performances of MNB with bagging might be due to the low variance of our data set, in which case bagging only makes a slight change in our accuracy.

When analysing the MLR models performance, the accuracy scores shows that the classifier does overfit slightly to the training data, with an accuracy score 8.7% above the result of the test data. This result is consistent across the other models and hyperparameter tuning has been done to decrease this difference where possible. The accuracy score result on the test data of 77.4% does however, still show that the MLR models performance is highest amongst the implemented models.

We also assess the performances of our models by F1 scores since there is no preference for either false positive or false negative in our scenario. F1 scores balance the importances of both precision and recall, hence we use it as a concise metric to compare the performances of our models. In the results, as expected, MNB models with and without bagging have the same F1 score, further confirming the previous similar accuracy results of MNB models with and without bagging. SVM model also has a low F1 score similar to its accuracy scores, which further justifies our removal of consideration as the final model. Also, for the

MLR model, we observe that it performs well with a 75% F1 score. While the MLR model performs better compared the other two in the F1 score aspect.

From the above analysis of the results within the evaluation matrix, the multinomial logistic regression model performs with the highest accuracy and F1 score and so we conclude that it is the most suitable learner to solve the task of text classification.

Future improvement

The model we generated using multinomial logistic regression works reasonably well to solve the task of article recommendation, however, our current methods could be extended to produce a higher performing model. An obvious next step to improve current model performance would be to implement different learners. To extend on our attempt at using bagging, we could implement a random forest classifier, and by extension, use boosting within the training of this model in order to weight the relevant features. Another classifier that could be attempted is a k-nearest-neighbour classifier, which labels the instances based on similarity to the previously seen data. This would have allowed for computational efficiency during training but would perform slowly at query time due to the size of the dataset.

As aforementioned, use of ensemble learning could be better utilised in future improvements to the task. More specifically, a heterogenous base classifier could be attempted to combine the benefits of multiple classifiers, such as those previously implemented (MLR, MNB, SVM). The results of these models could be aggregated through voting or averaging. The benefit of this implementation would be an improvement to classification accuracy and generalisation.

Consideration could also be given to the initial collection of the data within the articles. By having access to the full articles, including the titles and authors of the texts, features could have been more usefully extracted. This level of data mining could not occur within the constraints of the assignment, but could provide useful insights when performing text classification in the future.

Conclusion

This report has sought to outline the methods we implemented when traversing the task of text classification, a problem that becomes more important due to the ever increasing number of texts available on the internet. We implemented three key classifiers (MLR, MNB, SVM) in our attempt to develop a highly accurate model and through continuous feature engineering and parameter tuning we found multinomial logistic regression to be the most suitable classifier for the task. This process has been highly exploratory and throughout, has involved trial and error as we implemented a range of tuning

methods and parameters. This task has taught us the complexities involved in tuning models, and within this the importance of feature selection when handling large datasets, in order to improve computational efficiency. Overall this task has been insightful in learning the applications of classifiers within a text recommendation problem and we conclude that multinomial logistic regression is the best learner for the task.

Code

Please check out the .zip file

References

Huimin, Fan, Pengpeng, Li, Yingze, Zhao, Danyang, Li. 2018, 'An Ensemble Learning Method for Text Classification Based on Heterogeneous Classifiers' in *International Journal of Advanced Network, Monitoring and Controls*, 3(1), pp. 130-134.

Iglesias, E. L., Romero, R., Borrajo, L. 2015. 'A Linear-RBF Multikernel SVM to Classify Big Text Corpora', *BioMed Research International*, pp. 1-14.

Muller, Andreas C., Guido, Sarah. 2017, 'Chapter 7 Working with Text Data', in *Introduction to Machine Learning with Python – A Guide for Data Scientists*, editor Dawn Schanafelt, O'Reilly Media, California, pp. 323-356.

van der Maaten, Laurens., Hinton, Geoffrey, 2008, 'Visualising Data using t-SNE', *Journal of Machine Learning Research*, 9, pp. 2579-2605.

Zafra, Miguel Fenandez. 2019, *Text Classification in Python*, viewed 30 March 2020, < https://towardsdatascience.com/text-classification-in-python-dd95d264c802>.

Appendix

A1 – Article Distribution %

A2 – Top 10 most frequent words under bag of words methodology

A3 – Top 10 most highest TF-IDF words

Tables in here get completely messed up when pasted into the google doc, left them out for now.