School of Engineering and Applied Science (SEAS), Ahmedabad University

BTech(ICT) Semester VI: Machine Learning (CSE 523)
End Semester Project Report

News Article Classification

Yesha Ajudia
AU1841078

Kartavi Baxi
AU1841079

Harsh Kakasaniya
AU1841085

Vimarsh Soni
AU1841121

***Abstract*—This project is based on a supervised machine learning text classification model which would be able to predict the category of a given news article from the predefined set of categories. It uses a labelled dataset helping the algorithm to learn the patterns and correlations in the data. Data cleaning is used to ensure no distortions to the model. TF-IDF vectorizer is implemented using uni-grams and bi-grams as features, text formatted dataset is converted to numeric form which is then used to analyse the category of the given news articles. This helped us to analyse patterns in the data and gain valuable insights from it. Further, using the Multinomial Naive Bayes, as a machine learning classification model to predict the class of the test data and tuning the hyper-parameter, we were able to predict the class of test dataset with accuracy of the model being 98.43% and a log loss of 0.105. Hence, Multinomial Naive Bayes can be used as an effective classification method for the given dataset.**

***Index Terms*—Text Classification | Multinomial Naive Bayes | Laplace Smoothing | Hyperparameter Tuning | Vectorization | Supervised Machine Learning**

I. Introduction

Multiple subject labels are often used to report news articles. For example, in the context of sport, business, and world news a report on transfer ownership of the football club might be marked. Users prefer to read articles/news, based on their areas of interest. So applications and websites use different ways to sort the articles based on different categories that make the search comfortable. People can accurately recognize and give various noteworthy labels for an article, but can a machine learning framework deliver comparable findings?

The machine learning model for automated category-wise text classification could be used to identify topics of untracked news and make individual suggestions based on the user's prior interests [14]. Text classification offers a good framework for getting similar content with textual data processing without lacking its uniqueness [16]. Hence, this paper focuses on the classification of publicly available BBC news articles dataset among 5 predefined categories: Business, Entertainment, Politics, Sports and Tech, which can be helpful for a reader. The dataset consists of content of the article and its category, which can be visualised as a key-value pair with article being the key and category being the value. Label encoding is used to help the algorithm to learn the patterns for those occurrences of the words in the content. The classification model characterizes each article based on the features extracted using frequency distribution of words in the content. For optimization of the learning algorithm, hyper-parameter tuning is used to control the learning process. Every provided article will be analysed by its words list frequency and then evaluated to classify it in one of the 5 predefined categories.

II. Literature survey

The frequency distribution of every uni-gram and bi-gram gives the idea about the frequent words appearing in all categories. N-grams are a viable alternative to words as indexing terms in information retrieval. N-grams provide higher accuracy than a strawman system using raw words as indexing terms [3]. Machine learning techniques that have been actively explored for text classification include Naive Bayes classifier, K-nearest neighbor classifiers, support vector machine, neural networks [1]. Naive Bayes is a simple probabilistic classifier which works on the assumption of conditional independence between the features of a text document [2], which is the so-called "Naive Bayes assumption" [8]. Naive Bayes has been successfully applied to document classification in many research efforts [8]. When compared to the state-of-the-art algorithms for decision tree induction, instance-based learning, and rule induction on standard benchmark datasets, Naive Bayes is found to be sometimes superior to the other learning schemes, even on datasets with substantial feature dependencies [4]. It is based on the Bayes rule which says

$$P(c|d) = \frac{P(d|c)P(c)}{P(d)} \tag{1}$$

where $P(c|d)$, is the probability of document d to belong to class c, called the posterior probability, $P(d|c)$ is the likelihood and $P(c)$ is the prior. Naive Bayes classifies the document to the class which maximizes the posterior probability [2]. There are mainly two models for Naive Bayes classification, they are Multivariate Bernoulli Model and the Multinomial Model [5]. The Multivariate model has binary representation of features while the later represents the features with term frequency [2]. It was observed in [6] that Multinomial Naive Bayes performs better than Multivariate Bernoulli Naive Bayes. Multinomial Naive Bayes event model is more suitable when the dataset is large when compared to the Multi-variate Bernoulli Naive Bayes Model [7]. One systemic problem is that when one class has more training examples than another, Naive Bayes selects poor weights for the decision boundary [9]. There are many variants of Naive Bayes evolved, such as Complement Naive Bayes, Weight-normalized Complement Naive Bayes, Transformed Weight-normalized Complement Naive Bayes, which tackles with the problems arising out of the “Naive Bayes assumption” and imbalance in the classes of training examples. A central issue in language model estimation is smoothing, the problem of adjusting the maximum likelihood estimator to compensate for data sparseness [10]. A problem with Naive Bayes classifier is that it underestimates the likelihoods of words that do not occur in the data. A simple solution to this is called ”Laplace’s law of succession” or ”add one smoothing”, which adds a count of one (also called a ”pseudo-count”) to each word type [11]. The smoothing techniques not only avoid zero probability for unseen words, but also achieve an improved estimation for the whole model [12]. The smoothing techniques include Laplace smoothing, Dirichlet smoothing, Absolute Discounting. [13] concludes that Laplace smoothing is superior to Dirichlet smoothing and Absolute Discounting.

III. Implementation

The dataset is collected from [14], which is a publicly available BBC news dataset. It consist of 2225 samples of data, which are divided into 5 categories as mentioned earlier. The text dataset from different files was collected and a csv file was created for the same. The dataset is balanced which helps in reducing the chances of mis-classification. Data was then cleaned by converting the short forms to full forms i.e., converting $'d$ to $would$, $'ll$ to $will$, $can't$ to $cannot$, etc, removing all the characters other than alphabets from the text, removing unwanted spaces, and converting the text to lower case. Further, lemmatization is performed on the words, and stop words are removed. The inbuilt nltk library for stop words removal is used, but some of the words like $would$, $could$, $also$, etc, which are not removed through the inbuilt stop words list are added to the stop words list manually. The frequency distribution of uni-grams and bi-grams occurring in the cleaned text is then plotted to observe if a frequency based feature selection method could be implemented. The labels in the dataset are in the form of text, which need to be converted to numeric form, using label encoding. Hence, the next step performed is label encoding. It basically converts an array of unique classes to a numeric array of [0, number of classes - 1]. The data is then split into 80% training data and 20% test data, by selecting randomly from the dataset.

TF-IDF vectorizer is implemented using uni-grams and bi-grams as features, to convert the text features to numeric vectors, based on the frequency of a word occurring in a document. Multinomial Naive Bayes is used as a classifier, which can be modeled to predict the class of the test data. As mentioned earlier, Multinomial Naive Bayes is based on the Bayes rule, which calculates the posterior probability. The document is classified as the class which has the highest posterior probability. The likelihood in equation (1) is calculated

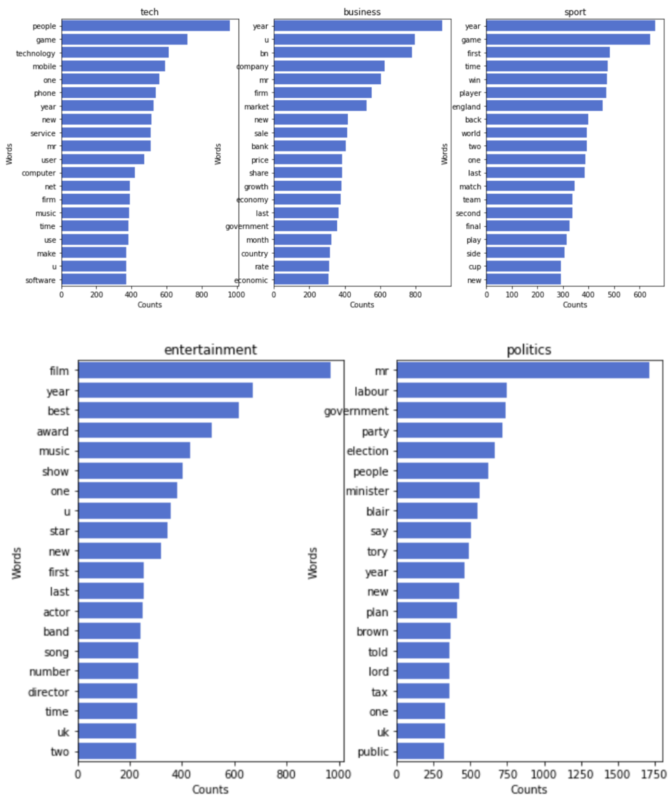


Fig. 1. Uni-gram Frequency

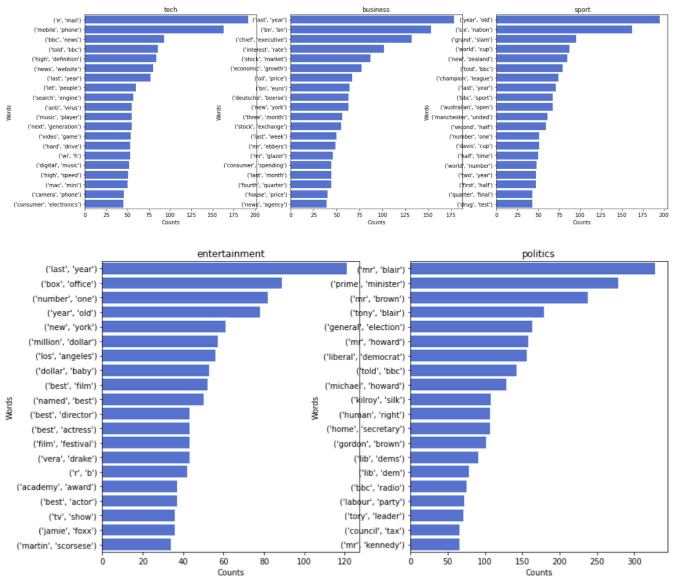


Fig. 2. Bi-gram Frequency

as:

$$P(d|c) = \gamma \prod_{t=1}^{|V|} P(w_t|c_k)^{x_t} \quad (2)$$

where, $|V|$ is the vocabulary size, w_t is the word, c_k is the k^{th} class, x_t is the frequency of word w_t in document d and $\gamma = \frac{n!}{\prod_{t=1}^{|V|} x_t!}$ (n is the total number of words in document d) is the normalization term, which is generally not considered as it does not depend on the class. $P(w_t|c_k)$ is the relative frequency of w_t in documents of class k with respect to the total number of words in documents of that class [11]. The class prior P(c) can be estimated by dividing the number of documents belonging to class c by the total number of documents [15].

A limitation of this idea is that while computing $P(w_t|c_k)$, a zero count might lead to zero probability, as the likelihood takes into account the product of probabilities. If a word does not occur in a class in the training samples, it does not imply that, that particular word will not occur in any of the texts of that class. Hence, to solve this issue, the Laplace's law of succession comes in. It adds one count to each word. Therefore, the equation of $P(w_t|c_k)$ becomes:

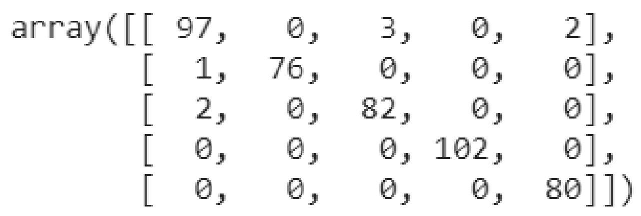
$$P(w_t|c_k) = \frac{1 + f_k(w_t)}{|V| + \sum_{i=1}^{|V|} f_k(w_i)} \quad (3)$$

where, $f_k(x)$ is the count of word x in all the training documents belonging to class k.

The Laplace's law of succession adds a count of one to each word. This count is known as the hyperparameter of Multinomial Naive Bayes, which can be tuned to get higher cross-validation score and hence higher accuracy. Hyperparameter is parameter that controls the form of the model itself. Different classifiers may have multiple hyperparameters. However, the more the hyperparameters to be tuned, the slower the tuning process is. Here, we tune the hyperparameter of Multinomial Naive Bayes known as the "smoothing parameter", α. While implementing Multinomial Naive Bayes, a list of values of α is used to find the best one. Hyperparameter tuning uses a Grid Search method, which exhaustively considers all parameter (α) values provided, to find the best possible value of α. Here, a 3-fold cross validation strategy is used to evaluate the performance of the model, and the best value of the α is returned. Then the classifier is made to fit the training data with the best value of hyperparameter obtained. The classes and their probabilities on the test data are predicted, and confusion matrix and classification report are generated.

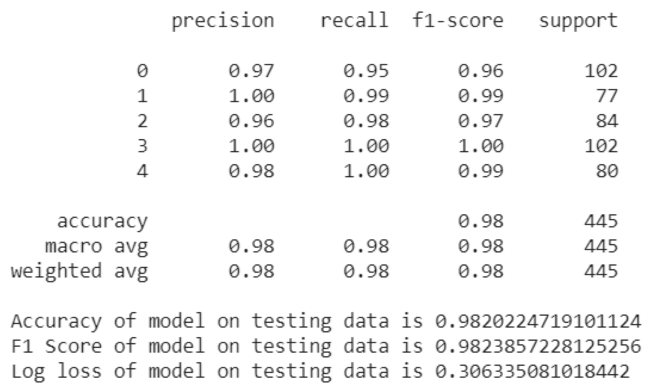
IV. Results

The samples in test data are assigned a category based on maximum posterior probability. There was a slight (can be considered nearly negligible) increase in accuracy and a decrease in log loss obtained after implementing hyperparameter tuning. The accuracy increased by nearly 0.2% and the loss value decreased by 0.2. Fig. 5 shows the training and cross validation accuracy for different values of α. Here, the x-axis represents the log_{10} values of parameter α. From Fig. 5, it can be inferred that the best value of α in this case would be 10^{-2} = 0.01, where both the accuracy are high and have the minimum difference between them, which is also observed in Fig. 6.



```
array([[ 97,   0,   3,   0,   2],
       [  1,  76,   0,   0,   0],
       [  2,   0,  82,   0,   0],
       [  0,   0,   0, 102,   0],
       [  0,   0,   0,   0,  80]])
```

Fig. 3. Confusion Matrix without hyperparameter tuning, where rows indicate actual value and columns indicate predicted value



```
              precision    recall  f1-score   support

           0       0.97      0.95      0.96       102
           1       1.00      0.99      0.99        77
           2       0.96      0.98      0.97        84
           3       1.00      1.00      1.00       102
           4       0.98      1.00      0.99        80

    accuracy                           0.98       445
   macro avg       0.98      0.98      0.98       445
weighted avg       0.98      0.98      0.98       445

Accuracy of model on testing data is 0.9820224719101124
F1 Score of model on testing data is 0.9823857228125256
Log loss of model on testing data is 0.306335081018442
```

Fig. 4. Classification report without hyperparameter tuning

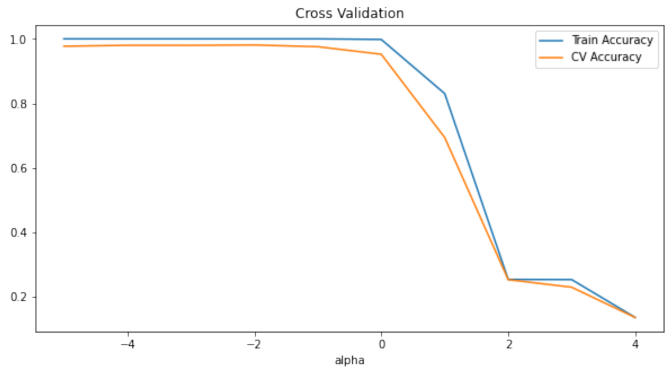
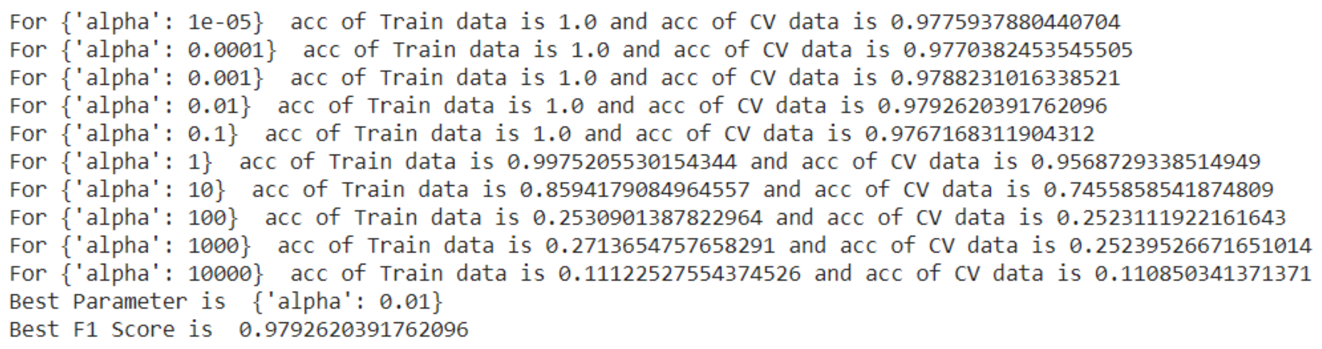


Fig. 5. Training and Cross Validation Accuracy

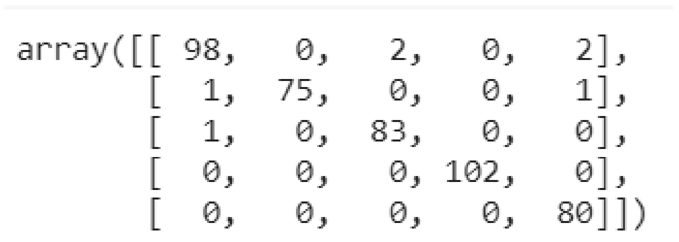
V. Conclusion

This paper explains the detailed work done for news article classification which includes all the necessary steps, like data collection and pre-processing, cleaning of data, observing frequency distribution of uni-grams and bi-grams, label encoding, splitting of train and test data, implementation of Multinomial



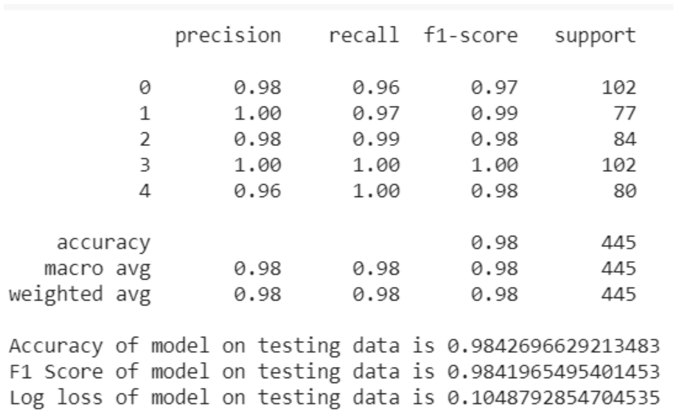
```
For {'alpha': 1e-05}  acc of Train data is 1.0 and acc of CV data is 0.9775937880440704
For {'alpha': 0.0001}  acc of Train data is 1.0 and acc of CV data is 0.9770382453545505
For {'alpha': 0.001}  acc of Train data is 1.0 and acc of CV data is 0.9788231016338521
For {'alpha': 0.01}  acc of Train data is 1.0 and acc of CV data is 0.9792620391762096
For {'alpha': 0.1}  acc of Train data is 1.0 and acc of CV data is 0.9767168311904312
For {'alpha': 1}  acc of Train data is 0.9975205530154344 and acc of CV data is 0.9568729338514949
For {'alpha': 10}  acc of Train data is 0.8594179084964557 and acc of CV data is 0.7455858541874809
For {'alpha': 100}  acc of Train data is 0.2530901387822964 and acc of CV data is 0.2523111922161643
For {'alpha': 1000}  acc of Train data is 0.2713654757658291 and acc of CV data is 0.25239526671651014
For {'alpha': 10000}  acc of Train data is 0.11122527554374526 and acc of CV data is 0.110850341371371
Best Parameter is  {'alpha': 0.01}
Best F1 Score is  0.9792620391762096
```

Fig. 6. Parameter Calculation



```
array([[ 98,   0,   2,   0,   2],
       [  1,  75,   0,   0,   1],
       [  1,   0,  83,   0,   0],
       [  0,   0,   0, 102,   0],
       [  0,   0,   0,   0,  80]])
```

Fig. 7. Confusion Matrix with hyperparameter tuning, where rows indicate actual value and columns indicate predicted value



```
              precision    recall  f1-score   support

           0       0.98      0.96      0.97       102
           1       1.00      0.97      0.99        77
           2       0.98      0.99      0.98        84
           3       1.00      1.00      1.00       102
           4       0.96      1.00      0.98        80

    accuracy                           0.98       445
   macro avg       0.98      0.98      0.98       445
weighted avg       0.98      0.98      0.98       445

Accuracy of model on testing data is 0.9842696629213483
F1 Score of model on testing data is 0.9841965495401453
Log loss of model on testing data is 0.1048792854704535
```

Fig. 8. Classification report with hyperparameter tuning

Naive Bayes classification model, along with hyperparameter tuning. The hyperparameter tuning involves Grid Search for the best parameter, evaluated using k-fold cross validation (for k=3). The optimal value of the hyperparameter found is α = 0.01, which gives an accuracy of 98.43%. It is observed that as the value of α increases, the training and cross validation accuracy tend to decrease. The tuning was also tried with different values of k in k-fold cross validation; however the optimal value of α did not change, hence no change in the results were observed.

References

[1] Patra, Anuradha, and Divakar Singh. "A survey report on text classification with different term weighing methods and comparison between classification algorithms." International Journal of Computer Applications 75.7 (2013).

[2] Vijayan, Vikas K., K. R. Bindu, and Latha Parameswaran. "A comprehensive study of text classification algorithms." 2017 International Conference on Advances in Computing, Communications and Informatics (ICACCI). IEEE, 2017.

[3] McNamee, P., Mayfield, J. Character N-Gram Tokenization for European Language Text Retrieval. Information Retrieval 7, 73–97 (2004).

[4] Domingos, Pedro, and Michael Pazzani. "On the optimality of the simple Bayesian classifier under zero-one loss." Machine learning 29.2 (1997): 103-130.

[5] Aggarwal, Charu C., and ChengXiang Zhai. "A survey of text classification algorithms." Mining text data. Springer, Boston, MA, 2012. 163-222.

[6] Singh, Gurinder, et al. "Comparison between multinomial and Bernoulli Naive Bayes for text classification." 2019 International Conference on Automation, Computational and Technology Management (ICACTM). IEEE, 2019.

[7] Aghila, G. "A Survey of Naïve Bayes Machine Learning approach in Text Document Classification." arXiv preprint arXiv:1003.1795 (2010).

[8] McCallum, Andrew, and Kamal Nigam. "A comparison of event models for naive bayes text classification." AAAI-98 workshop on learning for text categorization. Vol. 752. No. 1. 1998.

[9] Rennie, Jason D., et al. "Tackling the poor assumptions of naive bayes text classifiers." Proceedings of the 20th international conference on machine learning (ICML-03). 2003.

[10] Zhai, Chengxiang, and John Lafferty. "A study of smoothing methods for language models applied to information retrieval." ACM Transactions on Information Systems (TOIS) 22.2 (2004): 179-214.

[11] Shimodaira, Hiroshi. "Text classification using naive bayes." Learning and Data Note 7 (2014): 1-9.

[12] He, Feng, and Xiaoqing Ding. "Improving naive bayes text classifier using smoothing methods." European Conference on Information Retrieval. Springer, Berlin, Heidelberg, 2007.

[13] Indriani, Fatma, and Dodon T. Nugrahadi. "Comparison of Naive Bayes smoothing methods for Twitter sentiment analysis." 2016 International Conference on Advanced Computer Science and Information Systems (ICACSIS). IEEE, 2016.

[14] British Broadcasting Corporation (BBC). "Insight - BBC Datasets." Insight Resources - BBC Datasets, BBC, 2006, mlg.ucd.ie/datasets/bbc.html.

[15] Kibriya, Ashraf M., et al. "Multinomial naive bayes for text categorization revisited." Australasian Joint Conference on Artificial Intelligence. Springer, Berlin, Heidelberg, 2004.

[16] "Text Classification: The First Step Toward NLP Mastery." Medium, 12 Sept. 2020, medium.com/data-from-the-trenches/text-classification-the-first-step-toward-nlp-mastery-f5f95d525d73.

VI. GitHub Link of Project

https://github.com/vimarsh-soni/News-Article-Classification