News Classification

What is the best Text Classification method to classify News into the correspondent category?

|  |  |  |  |
| --- | --- | --- | --- |
| Carla McKeon  Student ID: 21267441  MSc in Computing (Full Time)Dublin City University [carla.mckeon2@mail.dcu.ie](mailto:carla.mckeon2@mail.dcu.ie) | Claudio Rocha  Student ID: 20213637 MSc in Computing (Part Time)Dublin City University [claudio.rocha2@mail.dcu.ie](mailto:claudio.rocha2@mail.dcu.ie) | Roshni S. Ramrakhyani  Student ID: 20211305 MSc in Computing (Full Time)Dublin City University [roshniramrakhyani2@mail.dcu.ie](mailto:roshniramrakhyani2@mail.dcu.ie) | Deepanshu Narwat  Student ID: 20211306 MSc in Computing (Full Time)Dublin City University [deepanshu.narwat2@mail.dcu.ie](mailto:deepanshu.narwat2@mail.dcu.ie) |

*Abstract*— The importance of text classification is becoming more apparent with the ever-growing size of documents available on the internet. Finding information of personal interest from a wealth of available news articles has become a difficult task for the regular internet user. Thus, the purpose of this research is to classify news articles into different categories so that a user can identify articles given their interest in certain categories. For this, we compare and contrast a number of different classification methods. We aim to beat the benchmark accuracy result reached on our dataset with the state-of-the-art RoBERTa model.

Text classification was performed on news articles from New York Times. The data was cleaned using standard techniques and data was explored to extract important features. The articles were from 24 different genres. A series of different classification algorithms were used, from conventional Naive Bayes to sophisticated RNN. Using all these classifiers helps us compare their performance for the said task and achieve comparable scores with the benchmark Bert model.

Keywords—classification, lemmatization, imbalanced classification, tf-idf embedding, glove embedding, SVM, RNN, KNN, random forest, precision, recall, support

# Introduction

Text classification is the process of assigning text documents to one or more predefined categories. This allows users to find desired information faster by searching only the relevant categories, rather than the full information space. The importance of text classification is even more apparent when the information space is vast, diverse, and ever-growing in size such as on the Internet. Finding information of personal interest from the wealth of available news articles has become a difficult task for the regular Internet user.  Thus, the purpose of this research is to classify news articles into different categories so that the user can identify articles given their interest in certain categories. Our corpus of news articles is taken from the New York Times and spans 24 categories such as Health, Real Estate, Food, Sports, etc.

In this paper, we focus our efforts on using traditional machine learning classification techniques such as Naïve Bayes [12], KNN [3] and SVM [4] to reach the benchmark accuracy achieved on this dataset by the deep learning BERT model outlined in [8]. Modern deep learning models consume a massive amount of energy given their dependence on high computational resources. Building and deploying these models require large datasets, millions of parameters, hours of computation time, a tremendous amount of energy and thus high carbon emissions. In 2018, the BERT model [2] achieved best-in-class NLP performance after it was trained on a dataset of 3 billion words. BERT, a transformer-based model is the most state-of-the-art NLP system to date. Before transformers, NLP relied on RNN [7] and LSTM [5]. One of the major difficulties when applying Transformers to applications is that it requires more complex configurations (e.g., optimizer, network structure, data augmentation) and a lot of additional GPU. Thus, we will attempt to reach this benchmark set out by [8] using a RNN approach.

# Related Work

Reference [8] propose the dataset, N24News, that we are using in this project. They set the benchmark for text classification at 88.86 using word2vec embeddings and RoBERTa [22], which is one of the current state-of-the-art models used for text classification. Their analysis found that using the body of the article as the input feature was most effective. While the key focus of this paper was on analyzing the images associated with a news article in order to demonstrate how images and text can be combined into a multimodal model for enhanced classification. We took their text classification results as benchmarks for our analysis of the body of text in the articles.

Reference [9] compare and contrast the performance of a number of word embedding methodologies from context-independent like GloVe, FastText and word2vec to context-dependent ELMo and BERT for use in both CNN and BLSTM classification techniques. Their experiments were trialed on a number of different text classification datasets. In particular, for the multi-label classification problems they found that GloVe-300 gave the highest reported accuracy and F1 score. This paper concludes that when selecting GloVe for a general classification problem, it is best to use its 100-200 (6B tokens) dimensional pre-trained embeddings.

Reference [10] produces an intuitive comparison between TF-IDF and Word Embeddings and helps us understand the importance TF-IDF embeddings in classification tasks. During the modelling phase TF-IDF was found to be the most useful representation of text data and hence used as a baseline approach.

Reference [11] uses Multinomial Naïve Bayes algorithm for sentiment analysis of tweets. This is very similar to our task as both use text mining and classification based on vectors. Reference [12] was studied to understand the difference between Naïve Bayes and Multinomial Naïve Bayes algorithms. The concepts of using Gaussian Naïve Bayes as a classifier was taken from [13] where the authors use GNB for multivariate pattern analysis. The main task consisted of classifying searchlight signals or neural activity across large sets of small regions. It explains in-depth how GNB assumptions can be helpful in many cases. The results posted show similar accuracy to SVMs. Reference [14] was studied to understand the difference between various Naïve Bayes techniques and practical implementation of GNB.

Reference [4] uses SVMs for similar text classification and sheds light on how to use word-embedding based techniques with SVM classifiers. Study of [15] was done to understand the difference between binary and multi-class SVM classifiers.

# Data Description

## Dataset Collection

The N24News is extracted as outlined in [8] and is taken from the New York Times. The New York Times is an American daily newspaper that publishes worldwide news on various topics every day. The dataset was initially obtained through an API provided by the New York Times to all links published from 2010 to 2020. All news on the site belongs to one of 24 categories, our collection contains up to 3000 samples for each category. The collection of articles in our set is over 60K. The amount in each category is shown in figure 3. We found this dataset on GitHub.com [23] and downloaded the related articles via the URL links provided in this file.

## Exploratory Data Analysis

For the exploratory Analysis, 6 different aspects were considered:

Diagram, application

Description automatically generated

1. Features

The Dataset contains 8 columns with different features and it’s important to understand what exactly they represent, and which ones would be more useful to perform text classification. Here is a brief description of each one of them:

* Section: Category associated to the News Article
* Abstract: Condensed presentation of the substance of the Body’s News Article
* Article Url: Link where the article is available
* Body: News Article
* Caption: Descriptive heading of the News Article
* Headline: Title of the News Article

1. Number of entries

The number of entries is an important aspect to consider verify because it allows us to understand what the most adequate text classification techniques could be to perform on this data. This dataset contains 61 198 entries and could be used with traditional text classification techniques as well as with more sophisticated ones such as neural networks.

1. Null Values

Table

Description automatically generatedDetermining if there are any null values it’s extremely important before training any model, because it could lead to incorrect results. The only feature containing Null Values is “Caption” and can be considered irrelevant for this analysis because the main goal is to use the “Body” to accurately predict the corresponding category. The following results were obtained:

*Figure 1. Null Values for each feature.*

1. Unique Values

Determining the Unique Values is a particularly important aspect in this specific case, to be able to see the number of different Sections contained in this dataset. This parameter tells us how many different labels our model is going to have, train and predict, with this dataset containing 24 unique values for the feature “Section”.

1. Data Types

Verifying the datatype of each column is helpful when preprocessing the data, because it would allow us to convert to the right format and adjust it depending on our necessities. The Figure below shows that all the features are “object” pandas data type, and it would not be necessary to make any change before Table

Description automatically generatedpre-process of the data.

*Figure 2. Data types of various features.*

1. Balanced vs Imbalanced

This is a particularly important aspect to verify especially because the outcome will determine which one is the best evaluation metric to the problem. The Figure below shows the number of Articles per Section and is possible to verify that the majority of the Categories have 3000 articles associated and therefore could be considered a balanced Dataset.

Table

Description automatically generated

*Figure 3. Number of articles per Section*

## Data Cleaning

For the initial Data Cleaning / Processing, the 5 following processes were followed:

A screenshot of a phone

Description automatically generated with medium confidence

Convert to Lower Case: The first step was to convert all the words into lower case. This process would avoid that the system interprets the same word (ex: “Book” and “book”) as 2 different words.

Remove Stop Words / Special Characters: Usually stop words occur in abundance and have no relevant information to be used in the classification model. In relation to the Special Characters, they are not normally displayed in article News and provide no information to the model.

Tokenization: The Tokenization process is very common on NLP models and breaks the raw text into words (tokens). These tokens help to understand the context and to develop the text classification model.

Lemmatization (POS tagging): Lemmatization converts the word into its base form and Part-of-speech tagging is the process of allocating a tag to each word (token) based on their part of speech (noun, verb, etc). These 2 processes combined help to give a context to each token and would help the model to have more precise results than using Stemming.

# Methodology

Once the data cleaning was done, we started with the feature engineering. We used two different techniques for generating embeddings for the classification task – TF-IDF and GloVe embeddings. After generating these embeddings, we used them in various classifiers.

## Feature Engineering

1. TF-IDF Embeddings

Word embedding techniques are used to represent words mathematically. There are many different techniques available like OneHot encoding, Word2Vec, FastText and TF-IDF. For this particular assignment we went with TF-IDF technique because of high level sophisticated vectors produced by it. TF-IDF is a statistical method used to ascertain the mathematical significance of any word within a document. It consists of two parts- the ‘term frequency’ and the ‘inverse document frequency’. The value or significance of any word is calculated by multiplying these 2 terms and getting the tf-idf. The ‘term frequency’ is simply the ratio of number of target term in a document to the total number of terms in that document. In addition to that we calculate the ‘inverse document frequency’ by taking the logarithm of the ratio of total number of documents to the number of documents that contain the target word. These 2 terms are multiplied to get the final tf-idf term for every word. The significance of this calculation is that any word that is very common and appears in every other document in the corpus gets a lower score than a word that is common only to a particular document. This technique can further be improved by removing any stop-words from the corpus beforehand.

The tf-idf vectorization uses a similar approach to OneHot encoding but value corresponding to any word is its tf-idf and not 1. This technique proves very useful in our task as during training the model learns the importance of any word in a particular article by its corresponding tf-idf score. So, for instance, if any article contains a word ‘democracy’, we have a higher chance of classifying the article as ‘politics’. For our classification task we used the following method:

* Removing all stop-words from the corpus using NLTK
* Removing all special characters and numbers from the corpus
* Using ‘TfidfTransformer’ to generate TF-IDF scores and corresponding vectors

1. GloVe Embeddings

Our experiments utilize the GloVe embeddings trained by (Pennington et al. 2014) on 6 billion tokens of Wikipedia and Gigaword 5. The GloVe model is trained using a matrix factorization technique on the word-context matrix. It builds a large matrix consisting of the co-occurrence of words, the purpose of this matrix is to tabulate how frequently words co-occur with one another in our corpus. The word embeddings are fine-tuned during training to improve the performance of classification. GloVe of 6B tokens with dimensionality size 100 was applied to our cleaned dataset, where there were 123,390 words in our word index successfully embedded.

## Classification

1. Decision Tree

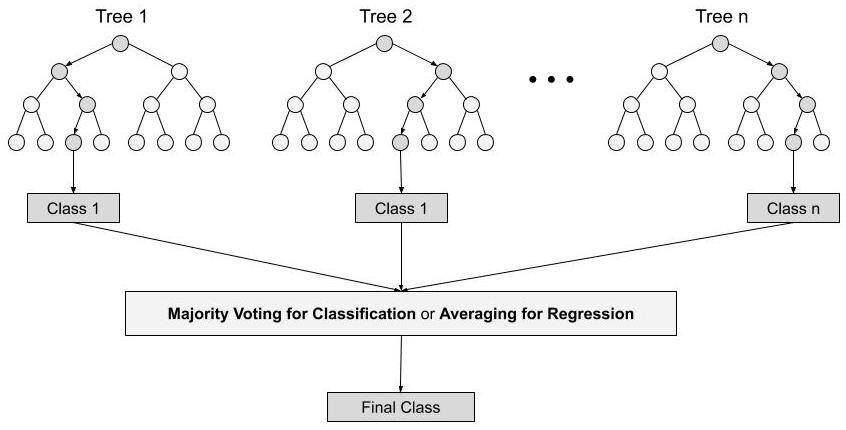
Decision tree is a supervised learning technique used for both classification and regression problems, mostly favored for classification problems. It is called a decision tree as the classifier resembles a tree like structure with internal nodes represent by features of the dataset, branch representing the decision rules and each leaf-node represents the outcome. Mainly the decision tree has two types of nodes decision nodes and leaf nodes. Decision nodes have many branches to it and are used to decide while the leaf nodes are the outcomes of the decisions made.

The key reason to use the decision tree as a classifier is that it is very easy to understand and works like a human brain to classify things into categories also the visual structure is easy to grasp. To predict a class of the given dataset the algorithm starts from the root node branching to multiple decisions possible. The algorithm compares the values of the root attributes and leads to a decision. These decisions branch further until the final decision is made. The root attributes continue to move down the tree to reach the leaf nodes or the final class decision and thus classifies an attribute. Here we have used the DecisionTreeClassifier() class from the sklearn.tree library.

1. Random Forest

Random Forest is an ensemble method for classification, regression and tasks that require decisions to be taken. The Random Forest is classification algorithm consisting or a large number of decision tress. This algorithm can handle continuous variables in case of regression and categorical variables in case of classification. The algorithm is a Bagging algorithm that creates a different training subset from sample training data with replacement and the final output is based on majority voting.

The method begins by collecting n random records from the dataset containing k number of records and then individual decision trees are constructed for each sample wherein each decision tree will generate an output. Averaging or counting the majority votes for each class from the outputs decision trees to calculate the final result.



*Figure 4. Illustration depicting classification using Random Forest classifier. (Abdulkareem, (2021))*

‘RandomForestClassifier()’ class is used from the ‘sklearn.ensemble’ library with n\_estimators=100 which is the default value meaning that the total number of decision trees for each record will be 100  ,criterion='entropy' for the information gain.

1. Logistic Regression

Logistic regression is used to find relationship between one dependent variable with one or more independent variables by estimating probabilities using a logistic regression equation. The Logistic regression algorithm is a classification algorithm for binary (e.g.- tumors malignant or benign.) or multi-linear function classification (e.g.- cats, dogs, goats, etc.).  It uses a complex cost function ‘sigmoid function’ also known as the ‘logistic function. The hypothesis of the logistic regression limits the cost function between 0 and 1. For an input passed in the prediction function, probability scores expected is between 0 and 1.

To classify the articles into one of the defined 24 classes we have used LogisticRegression() class from the sklearn.linear\_models library with max\_iter=1000. In other words, to converge the gradient descent will take maximum of 1000 steps.

1. Multinomial Naïve Bayes

Multinomial Naïve Bayes (MNB) classifier is a conventional approach to classification problems. It has been used for many similar semi-automated classification tasks such as spam mail classification. Although MNB is based on the same assumptions and principles as Naïve Bayes (NB) classifier, there is an important difference between NB and MNB - the NB classifier estimates probability of any class membership using Gaussian kernels, thus choosing the class, whose occurrence frequencies of the various packet sizes match best with the observed values in the test instance. One the other hand, MNB operates on the frequency distribution of all packet sizes at once.

The Naive Bayes method is a strong tool for analyzing text input and solving problems with numerous classes. Because the Naive Bayes theorem is based on the Bayes theorem, it is necessary to first comprehend the Bayes theorem notion. The Bayes theorem, which was developed by Thomas Bayes, estimates the likelihood of occurrence based on prior knowledge of the event's conditions. When predictor B itself is available, we calculate the likelihood of class A. It's based on the formula below:

P(A|B) = P(A) \* P(B|A)/P(B).

We used ‘MultinomialNB()’ method from the ‘sklearn’ library. The overall accuracy while testing was found to be 80%. The highest F1 score for a class was 0.92 with a support of 365 data points.

1. Gaussian Naïve Bayes

GNB is one of the simplest classification algorithms. It consists in assigning the label of the class that maximizes the posterior probability of each sample, under the assumption that the training data contributions are conditionally independent and obey a Gaussian or Normal distribution. Gaussian Naive Bayes supports continuous valued features and models each as conforming to a Gaussian distribution. This model can be fit by simply finding the mean and standard deviation of the points within each label, which is all what is needed to define such a distribution. Once the mean and standard deviations are found, at every data point, the z-score distance between that data point and each class mean is calculated to classify it into a particular group.

We used ‘GaussianNB’ method from the ‘sklearn’ library for classifying our news data.

1. K-Nearest Neighbors (KNN)

K-nearest-neighbour is a technique used for classification and regression prediction model. It is preferred for classification model due to its ease for interpretation and time complexity. The KNN algorithm assumes that the similar things fall into close proximity i.e., similar things are close to each other in the vector space. K-nearest-neighbour as its name suggests, classifies the products in the same category if the distance between then is smallest. Distance is calculated using Euclidean distance (also known as straight line distance), cosine similarity, etc preferably depending upon the problem we are solving. The distance between the test data and each row of the training data is calculated. Distance so calculated is arranged in sorted manner from smallest to largest, the top k rows from the sorted data is the class of these rows. The value of K is chosen so that it reduces the number of errors usually by hit-and trial method, keeping different value of k in the KNN algorithm to find the most appropriate k-factor.

We used KNeighboursClassifier() class of sklearn.neighbours library with the default k-value as 5 as the classifier implementing the K-nearest-neighbour. The classification is defined in numbers instead of the category name. KNN Classification report generated on the test data has no obvious patterns.

1. Support Vector Machines (SVMs)

Support Vector Machine or SVM is a supervised learning algorithm used for various tasks like classification, regression, and outlier detection. SVM classifier tries to find the best line that separates different classes. It divides an n-dimensional space with n features into two separate regions representing the different classes. The separation is done in a way that the hyperplane has the largest distance between the training vectors of different class, called ‘support vectors’. This hyperplane is called the Optimal Separating Hyperplane (OSH). Because of this approach SVM classifiers can be expected of generalizing or predicting more accurately on unseen data points. Moreover, since SVM classifier uses only few training points that lie at the edge of the class distribution as support vectors, it can produce better results even with smaller amount of training sets. The same cannot be said for any conventional classifier as they aim to minimize training errors and hence require a large training dataset.

Chart, scatter chart

Description automatically generatedChart, scatter chart

Description automatically generated

*Figure 5. Illustration showing different multi-class classification techniques in SVMs (One-to-One and One-to-All)*

SVM classifiers are generally used as binary classifier. However, the same binary approach can be extended for getting multi-class classifications. For the classification task for this assignment, we used a multi-class SVM to classify all the different classes in one go.

1. Recurrent Neural Network (RNN)

First proposed by [5], recurrent neural network (RNN) models with long short-term memory (LSTM) units have been adopted to reflect the fact that time sequence is an important property in text analysis. RNNs are successfully able to identify the temporal relationship across text documents. They do this by introducing an adaptive gating mechanism, which determines to which degree to keep the previous state and memorize the extracted features of the current data. While LSTM units add extra functions that mimic how humans think while speaking, given a sequence S = {x1, x2, . . . , xl}, where l is the length of input text, LSTM will process it word by word. However, LSTM can only read the information in one way limiting its power.

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
|  |  | **Table 1. Confusion Matrix** |  |  |
|  |  | Predicted as Positive | Predicted as Negative |  |
|  | Actually Positive | True Positives (TP) | False Negatives (FN) |  |
|  | Actually Negative | False Positives (FP) | True Negatives (TN) |  |

Bi-directional LSTM (BLSTM) proposed by (Schuster and Paliwal, 1997) has been proposed to replace uni-directional LSTM, further improving speech recognition accuracy. It introduces a second hidden layer, where both hidden connections are allowed flow in opposite temporal order. Meaning that the content is both propagated forward and backward through the network.

The neural network was built on Keras which is a deep learning API that runs on top of the machine learning platform TensorFlow. CuDNNLSTM is a fast LSTM implementation that can only be run on GPU. The model has two bidirectional

CuDNNLSTM(128) layers which have a hyperbolic tangent activation function, the final dense layer is a softmax, followed by a sparse categorical-cross-entropy loss function as we have more than two labels and a rmsprop optimizer. 10 epochs are run, as with additional epochs the validation loss begins to rise.

# Evaluation

Diagram

Description automatically generated with medium confidenceAccuracy is an evaluation metric defined by the number of correct predictions divided by the total number of predictions, and is given by the following formula:

According to [16], this measure is the most intuitive and common metric to evaluate the performance of classification predictive models but could be misleading in situations where the categories are unbalanced. Normally, machine learning models are biased towards the biggest categories and in this scenario, it would be possible to achieve high accuracy regardless of having limited skill over the minor classes.

As mentioned previously, our dataset can be considered balanced because most of the News Categories have 3000 articles associated, and for that reason “Accuracy” would be the most adequate measure to compare the different predictive models.

When comparing categories our problem transforms into an imbalanced class distribution. The results of the training samples can be categorized into four groups and denoted in the confusion matrix given in Table 1. F1-Measure measures only the performance of the positive class by considering Recall (R) figure 2 and Precision (P), figure 3. The F1-score is an average of these two, see figure 4.

# Results

The results obtained with the different text classification methods can be seen on the Figure 6 and show that the SVM classifier achieved the best performance among all with an accuracy of 0.86. This is very close to the benchmark score achieved in [2] using Bert model. A Similar high accuracy score of 0.85 was achieved using Logistic Regression.

The Figure 7 shows the average results per Section and is possible to verify that the ones obtaining best predictive results are “Dance” (0.89) and “Sports” (0.89) and the ones with the lowest scores were “Style” (0.32) and Well (0.32).

Table

Description automatically generated

Table

Description automatically generated*Figure 6. Number of articles per Section*

*Figure 7. F1 scores per Section*

# Conclusion

The results obtained in this task showed that traditional methods such as SVM (0.86) and Logistic Regression (0.85) obtained very close scores to the Bert Benchmark (0.88), while Decision Tree (0.60) and Gaussian Naïve Bayes (0.65) classifiers were the ones obtaining the worst results. When comparing the results across the different Sections, it’s possible to see the “Style” and “Well” were the ones obtaining the worst results. “Style” obtained poor results because of the similarity to the “Fashion” section, having more than 40% of the predictions being allocated to the latest Category. The Section “Well” had significantly less data (681 entries) to train the model, and that could explain the poor results. On the other hand, “Sports” and “Dance” obtained the best results with specific words such as “dance”, “dancer”, and “ballet” or “basketball”, “football” and “playoff” appearing very often and helping to achieve the highest results.

Our results fall short of the benchmark by just 2%. The advantage of SVM and other conventional machine learning methods over deep learning approaches such as the one used to generate the benchmark is that they are faster to employ and do not require GPU. Our SVM model ran in under one hour, while the second-best Logistic Regression took approximately 10 minutes and had an accuracy of 3% under the benchmark. From an environmental standpoint, there is evidence in this report to suggest that the time and energy consumption efficiencies gained with such methods outweigh the loss in accuracy.

##### References

1. S. Karita et al. "A comparative study on transformer vs rnn in speech applications." 2019 IEEE Automatic Speech Recognition and Understanding Workshop (ASRU). IEEE, 2019.
2. J. Devlin, M. Chang, K. Lee, K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding." arXiv preprint arXiv:1810.04805 (2018).
3. Shah, K. et al. (2020) “A comparative analysis of logistic regression, random forest and KNN models for the text classification,” Augmented human research, 5(1). doi: 10.1007/s41133-020-00032-0.
4. Kilimci, Z. H. and Akyokus, S. (2018) “Deep learning- and word embedding-based heterogeneous classifier ensembles for text classification,” *Complexity*, 2018, pp. 1–10. doi: 10.1155/2018/7130146.
5. S. Hochreiter, and J. Schmidhuber, "Long short-term memory." Neural computation 9.8, 1997.
6. J. Pennington, R. Socher, and C.D. Manning, "Glove: Global vectors for word representation." Proceedings of the 2014 conference on empirical methods in natural language processing, 2014.
7. M. Schuster, and K.K. Paliwal, "Bidirectional recurrent neural networks." IEEE transactions on Signal Processing 45.11, 1997.
8. Z. Wang, X. Shan and J. Yang. "N24News: A New Dataset for Multimodal News Classification." arXiv preprint arXiv:2108.13327, 2021.
9. C. Wang, P. Nulty, and D. Lillis, “A comparative study on word embeddings in deep learning for text classification,” Proceedings of the 4th International Conference on Natural Language Processing and Information Retrieval, 2020.
10. Dess, D. and Riboni, D. “TF-IDF vs Word Embeddings for morbidity identification in clinical notes: An initial study”, unpublished.
11. Muhammad A., Kamran A. M., Abdul A. J., Saleemullah M., Anees A., Multinomial Naive Bayes Classification Model for Sentiment Analysis, unpublished
12. Herrmann, D., Wendolsky, R. and Federrath, H. “Website fingerprinting: Attacking popular privacy enhancing technologies with the multinomial naïve-bayes classifier,” in Proceedings of the 2009 ACM workshop on Cloud computing security, 2009
13. Marlis O. O., Agustin Lage C., Giancarlo V., Rainer G., Mitchell V. S. “Fast Gaussian Naïve Bayes for searchlight classification analysis”, unpublished.
14. Majumder, P. “Gaussian Naive Bayes, OpenGenus IQ: Computing Expertise & Legacy”, 2020.
15. Mathur, A. and Foody, G. M. (2008) “Multiclass and Binary SVM Classification: Implications for Training and Classification Users,” IEEE geoscience and remote sensing letters : a publication of the IEEE Geoscience and Remote Sensing Society, 5(2), pp. 241–245. doi: 10.1109/lgrs.2008.915597
16. Jason, F., “Text Classification for Records Management”, Monash University, Australia
17. Vimala, B., and Ethel, L. Y., “Stemming and lemmatization: A comparison of retrieval performances”, Member, IACSIT
18. Jonathan, J. W., and Chunyu, K., “Tokenization as the initial phase in NLP, City Polytechnic of Hong Kong.
19. Moldagulova, A. and Sulaiman, R. B. (2017) “Using KNN algorithm for classification of textual documents,” in *2017 8th International Conference on Information Technology (ICIT)*. IEEE, pp. 665–671.
20. Swain, P. H. and Hauska, H. (1977) “The decision tree classifier: Design and potential,” IEEE transactions on geoscience electronics, 15(3), pp. 142–147. doi: 10.1109/tge.1977.6498972.
21. Abdulkareem, N. M. and Abdulazeez, A. M., “Machine Learning classification based on Radom Forest algorithm: A review,” International Journal of Science and Business, 5(2), pp. 128–142, 2021.
22. Liu, X. *et al.* (2019) “Improving multi-task deep neural networks via knowledge distillation for natural language understanding,” *arXiv [cs.CL]*. doi: 10.48550/ARXIV.1904.09482.
23. <https://github.com/billywzh717/N24News/blob/main/nytimes_dataset.json>