Cardiff School of Computer Science and Informatics

CMT307 Coursework 2 Group Project
Group 3

Text Categorisation

Authors:
C22086790
C22100089
C22052229
C22013271
C22106741
C1620070
C22093362

Supervisor:
Steven Silva Mendoza

Word count
3441

April 23, 2023

Contents

1 Introduction

Text categorisation, a subfield of natural language processing (NLP), involves automatically assigning pre-defined categories or labels to a given document or piece of text. It plays a vital role in various applications such as news classification, spam filtering, sentiment analysis, and content recommendation.[1] With the explosive growth of textual data on the internet, the importance of text categorisation has become paramount for businesses, governments, and individuals alike.[2] It has made significant strides in recent years, largely due to the advancements in machine learning techniques. Thanks to these advances, we now have more accurate and efficient algorithms for categorising large amounts of text data.

However, natural language is quite complex and presents its unique set of challenges for text categorisation. This requires the use of sophisticated algorithms and techniques that can handle the subtle nuances of language. To tackle these challenges, machine learning techniques such as supervised, unsupervised, and deep learning approaches have become the most popular methods for building text categorisation models. These techniques involve using statistical models, neural networks, and clustering algorithms to automatically identify patterns in the text data and map them to predefined categories.[3]

In this project, we will be using a dataset containing 20 newsgroups with approximately 20,000 documents to train three machine learning models to determine which newsgroup each document belongs to. The project focuses on the use of the following three machine learning models: Multinomial Naive Bayes (Multinomial NB), Long Short-Term Memory (LSTM) and Convolutional Neural Network (CNN). Additionally, we fine-tune the pre-trained Bidirectional Encoder Representations from Transformers (BERT) model to also evaluate one of the latest approaches to text classification.

2 Literature Review

2.1 Models

The analysis of text categorisation on multiple subjects and groups is becoming more researched. Barua, Sharif, and Hoque [4] (2021) looked at the six most popular machine learning techniques for text categorisation in the Bengali sports dataset. These machine learning techniques consisted of Multinomial Naïve Bayes (MNB), Random Forest (RF), Support Vector Machines (SVM), Logistic Regression, Decision Tree (DT), and Term Frequency-Inverse Document Frequency (TF-IDF). They used the Bengali news corpus, which consisted of 202830 unique words within 43306 news documents. The corpus was relating to multiple sport classes like football and tennis. Out of the six models, they discovered that the SVM model obtained the highest F1 score of 97.6 percent. A similar investigation was carried out by Daud, Ullah, Rehman, *et al.* [5] (2023), where the 2016-2018 Reuters data set was used to categorise online news articles. They used Stochastic Gradient Descent and K-Nearest Neighbour instead of DT and TF-IDF. Daud, Ullah, Rehman, *et al.* [5] also discovered that SVM performed better than other models with an accuracy of 85%. SVMs have high dimensional input space, meaning they can deal with many features [6]. However, a very noisy data set can affect the performance, leading to lower accuracy. Daud, Ullah, Rehman, *et al.* [5] demonstrated this in their article as SVM was the worst performing model without optimisation with an accuracy score of just under 65%. Therefore, having good pre-processing and hyper parameter tuning optimisation can combat this issue.

According to Li, Wang, and Xu [7] (2018), deep learning methods like long short-term memory (LSTM) and convolutional neural networks (CNN) displayed very good performance in classifying Chinese text. Deep learning models are capable of learning complex patterns and relationships in text, which can lead to improved performance on tasks such as sentiment analysis and natural language processing [8]. However, deep learning models require large amounts of training data and computational resources, which can be a challenge in some applications. Li, Wang, and Xu [7] created a model which incorporates two layers of LSTM and one layer of CNN to create the bi-directional long short-term memory CNN (BLSTM-C) model. They applied this model to three datasets and the highest accuracy achieved was 90.8%. This shows that deep learning models can also be successful in text classification.

Chen, Ye, Xing, *et al.* [9] (2017) created a CNN and RNN (recurrent neural networks) method for a multi-label text categorization dataset. They discovered that the size of the training dataset can have an impact on performance. If the data set was too small, it lead to overfitting. Therefore, their CNN and RNN worked best when trained on a large scale dataset.

2.2 Preprocessing

There are a variety of text pre-processing techniques that can help improve text categorisation datasets. HaCohen-Kerner, Miller, and Yigal [10] (2020) understood this, but wanted to know which method creates the greatest improvements. HaCohen-Kerner, Miller, and Yigal [10] discovered that for four benchmark text corpora (WebKB, R8, SMS Spam Collection and Sentiment Labelled Sentences), one combination of basic pre-processing would significantly improve the text categorisation. After applying Bayes Networks (BN), SMO (a variant of SVM) and RF models, it was discovered that stopword removal caused the greatest improvement in accuracy for three of the datasets. Stop words are viewed as words that are extremely common, yet insignificant [11] as they have no meaning. Removing the stop words helps decrease the dataset which in turn decreases the model training time and can help improve performance as only meaningful words are left [12].

More advanced pre-processing techniques are word lemmatisation and stemming. Stemming is a process that removes the last few characters from a word, whereas lemmatisation considers the context of the word and converts it back to its meaningful base form [13]. Stemming is an effective and efficient method in some cases - for example, removing the suffix of 'greeting' to be left with the word 'greet' - but it can lead to incorrect meanings and spelling in others - for example, the word 'caring' would become 'car' which has a completely different meaning. Lemmatisation, however, would return the word 'care' for 'caring'. Lemmatisation would provide more accurate results, but understanding the context of the word does come at a computational cost. If the dataset is large, it can be slow and expensive to calculate, whereas stemming would be completed quicker as it is just removing the last few characters.

3 Description of the Task/Dataset

3.1 Task

The 20newsgroups dataset is a collection of approximately 20,000 newsgroup documents, partitioned across twenty different news groups. The task is to classify each document into one of these twenty news groups, making it a multi-class text classification problem. The dataset was obtained from the scikit-learn library.

The dataset contains text data, which requires preprocessing before being fed into the machine learning models. We applied a series of preprocessing steps using a custom transformer called *TextCleanerPreprocessor*. For a detailed explanation of these preprocessing steps and their justification, please refer to the Preprocessing Choices and Justification subsection in the Methodology section.

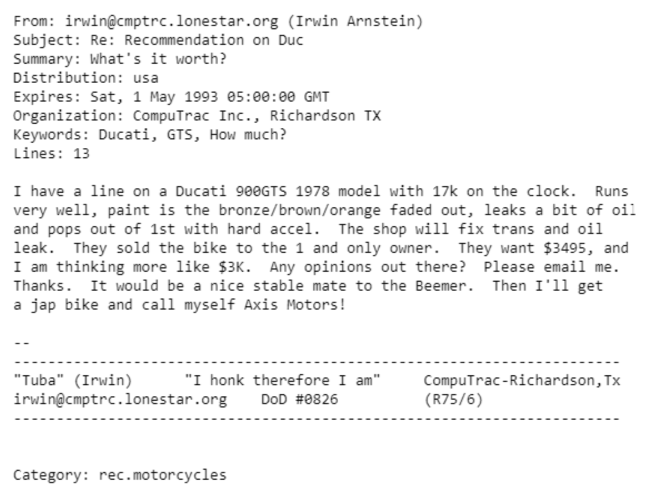
3.2 Exploratory Data Analysis

3.2.1 Analysing Raw Data

Referring to Figure 1, you can observe that the raw data in the 20newsgroups dataset contains numerous elements that are not useful for our model building process. These include headers, footers, email addresses, stop words, punctuation, and special characters. They introduce noise into the dataset and may lead to suboptimal performance of the model. It is essential to preprocess the data by filtering out these unnecessary elements, allowing the model to focus on the relevant features that contribute to the document categorisation task. By doing so, we can also reduce the dimensionality of the dataset and improve the efficiency and accuracy of our model.

3.2.2 Distribution of documents across categories

The number of documents in the training and test sets is shown in the visualisation of the distribution of documents across categories in the 20newsgroups dataset. The data is



```
From: irwin@cmptrc.lonestar.org (Irwin Arnstein)
Subject: Re: Recommendation on Duc
Summary: What's it worth?
Distribution: usa
Expires: Sat, 1 May 1993 05:00:00 GMT
Organization: CompuTrac Inc., Richardson TX
Keywords: Ducati, GTS, How much?
Lines: 13

I have a line on a Ducati 900GTS 1978 model with 17k on the clock.  Runs
very well, paint is the bronze/brown/orange faded out, leaks a bit of oil
and pops out of 1st with hard accel.  The shop will fix trans and oil
leak.  They sold the bike to the 1 and only owner.  They want $3495, and
I am thinking more like $3K.  Any opinions out there?  Please email me.
Thanks.  It would be a nice stable mate to the Beemer.  Then I'll get
a jap bike and call myself Axis Motors!

--
------------------------------------------------------------------------
"Tuba" (Irwin)      "I honk therefore I am"     CompuTrac-Richardson,Tx
irwin@cmptrc.lonestar.org    DoD #0826          (R75/6)
------------------------------------------------------------------------


Category: rec.motorcycles
```

Figure 1: raw data.

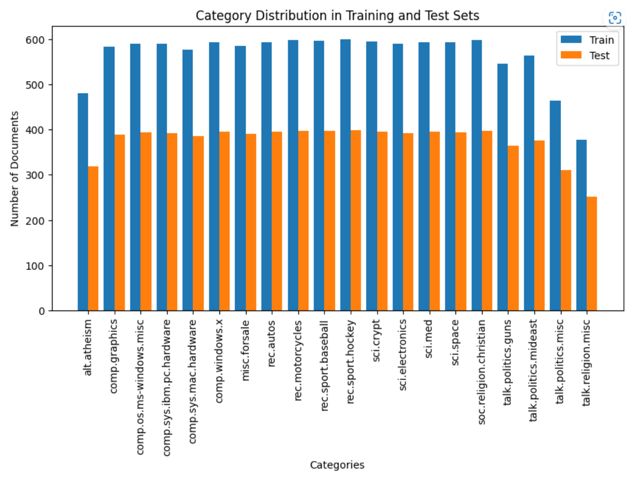


Figure 2: Category Distribution

well-balanced, with no obvious imbalance in document distribution between categories.

3.2.3 Visualising common words across all categories with word clouds

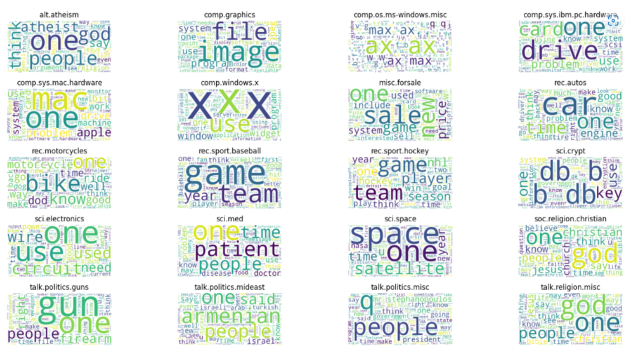


Figure 3: Word Clouds

Our analysis of the 20newsgroups dataset using word clouds (Figure 4) revealed several interesting trends in the most common words across different news groups. In general, words such as 'one', 'new', 'time', and 'people' were frequent across multiple news groups, indicating their widespread usage and relevance to a variety of topics. However, we also observed variations in the most common words across news groups. For example, in the 'alt.atheism' news group, words such as 'god', 'religion', and 'belief' were most frequent, reflecting a focus on philosophical and religious discussions. In the 'comp.graphics' news group, words such as 'file', 'image', and 'system' stood out, suggesting a focus on computer graphics and technical topics. Finally, in the 'rec.autos' news group, words such as 'car', 'engine', 'problem', and 'oil' were most common, highlighting discussions around automobile maintenance and troubleshooting.

3.2.4 N-gram Analysis

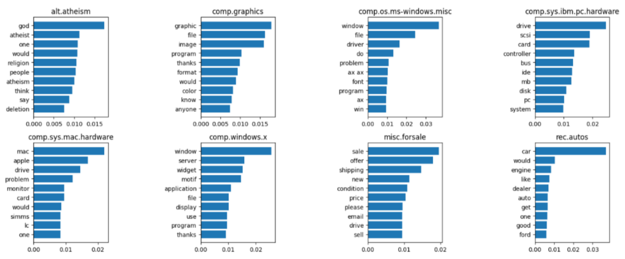


Figure 4: N-grams for different Categories

The bar charts in Figure 4, display the top 10 features with the highest average TF-IDF values for some of the twenty news groups, taking into account n-grams of size 1 to 3. Despite the fact that almost all most common n-grams are single words, it's important to consider all possible cases. The 2-gram "ax ax" is a top phrase in the comp.os.om-windows.misc news group. This emphasises the necessity of taking into account multi-word phrases when processing text input, as these n-grams may contain more particular and nuanced information, which can be useful in NLP applications.

From this n-gram analysis, we have concluded and summarised the most important points below.

- **Differences in vocabulary:** The top ten terms in each category are mostly unique, showing that the vocabulary used in each news group is different. This can help with the text classification task since variances in word usage can be used to distinguish across categories.

- **Term relevance:** The top terms with the highest TF-IDF values in each category seem to be relevant to their respective topics, suggesting that the TF-IDF technique effectively captures the most important words in each news group. This can help in feature selection and dimensionality reduction for text classification and NLP tasks.

4 Methodology

4.1 Model Selection Rationale

We selected various machine learning models for the 20newsgroups text categorisation task. These models were selected based on their suitability for textual data processing and the diversity of their fundamental mechanisms. This allowed us to compare their performance and understand their strengths and limitations.

Multinomial Naive Bayes was chosen as our baseline model because of its simplicity, computational efficiency, and well-established performance in text categorisation tasks. Furthermore, Long Short-Term Memory (LSTM) and Convolutional Neural Networks (CNN) were also implemented as deep learning models because they demonstrated promising results in text categorisation. LSTM is particularly good at managing sequential data, and CNN excels in learning local features. Finally, we chose BERT, a state-of-the-art transformer-based model, to utilise its pre-trained representations and fine-tuning capabilities to enhance classification performance potentially.

4.2 Preprocessing Choices and Justification

We performed preprocessing operations to prepare the text data for our machine-learning models to clean and standardise the text. These steps were selected due to their effectiveness in reducing noise and enhancing the quality of input data, which can lead to improved model performance:

1. **Removing headers, footers, and quotes:** These elements often contain unrelated information that does not contribute to the core meaning of the text. Removing them reduces noise, prevents models from relying on uninformative features, and helps focus on the main content of the discussion forum posts, ultimately improving model performance.

2. **Removing email addresses and URLs:** These elements are irrelevant to the text classification and can be considered noise.

3. **Handling contractions:** Expansion of contractions aids in preserving the correct meaning and form of words in a text.

4. **Removing non-alphabetic characters and lowercasing:** This step reduces the feature space and simplifies the text by avoiding case-sensitivity issues.

5. **Tokenising text into words:** Tokenisation is essential for preprocessing machine learning pipelines because it divides the text into meaningful units, enhancing subsequent analysis and the performance of text categorisation models.

6. **Removing stopwords:** Stopwords are common words with no significant meaning for classification tasks; removing them reduces data noise.

7. **Lemmatisation:** Lemmatising words to their root forms reduces the data's dimensionality by grouping similar words, which can enhance the performance of the models.

Furthermore, we applied identical preprocessing steps to all models to ensure a fair comparison. By applying consistent preprocessing to all models, we can gain a clearer understanding of the strengths and weaknesses of each model, as well as their relative classification performance.

4.3 Multinomial NB

4.3.1 Model Principles and Design

Multinomial Naive Bayes is a popular Bayes' theorem-based probabilistic classifier. Due to its simplicity, efficiency, and capacity to manage large feature spaces, it is particularly suited for text classification tasks.

4.3.2 Implementation Details and Pipeline

First, the text is transformed into a numerical representation using the Term Frequency-Inverse Document Frequency (TF-IDF) vectoriser. Then, the Multinomial Naive Bayes classifier is trained on the preprocessed data, and its performance is evaluated.

4.3.3 Hyperparameter Tuning

A grid search technique is employed to optimise the hyperparameters of the Multinomial Naive Bayes model, with a particular emphasis on the smoothing parameter (alpha). The optimal value for alpha is then determined, based on the model's performance on a validation set. Finally, the optimised model is evaluated with the test data to compare it with the non-optimised model.

4.4 LSTM

4.4.1 Model Principles and Design

LSTM is a form of recurrent neural network (RNN) designed to solve the vanishing gradient problem that concerns conventional RNNs. The ability of LSTMs to capture long-term dependencies in sequence data makes them suitable for classification tasks involving text. In the present study, we classify news articles into different groups using an LSTM model.

4.4.2 Implementation Details and Pipeline

After applying the standard preprocessor to the text data, tokenisation and padding is applied to ensure all input sequences have the same length. The LSTM model is then trained with different embedding dimensions (100, 200, and 300), and its performance is compared. In addition, we experiment with one other pre-trained GloVe embedding (Common Crawl 840B) to compare and potentially enhance model performance.

4.4.3 Hyperparameter Tuning

Using a grid search strategy, the LSTM models' hyperparameters are optimised, including the number of layers, hidden units, and learning rate. The model's efficacy on a validation set directs the choice of optimal hyperparameter values.

4.5 CNN

4.5.1 Model Design Principles

CNNs are commonly associated with image processing, but they have also proven to be effective in text classification. Their ability to identify local data patterns through convolutional layers makes them suitable for the project task. A CNN model architecture is implemented and evaluated to predict the newsgroup category and obtain an optimised version after hyperparameter tuning.

4.5.2 Implementation Details and Pipeline

To ensure equal sequence lengths, we additionally preprocess the previously preprocessed text data using tokenisation and padding. The data is then used to train the CNN model, which consists of the following layers:

- A layer containing 300-dimensional word embeddings.
- A dropout layer with a dropout rate of 0.2.
- A 1D convolutional layer with 128 filters, a kernel size of 3, a ReLU activation function, and a stride of 1.
- A global max-pooling layer for dimensionality reduction.
- A dense hidden layer with 250 units, ReLU activation function, and a 0.2% dropout rate.
- A dense output layer comprised of 20 units equal to the number of classes, followed by a softmax activation function.

The model is compiled using the Adam optimiser and sparse categorical cross-entropy loss. An early stopping strategy is employed to prevent overfitting with the patience parameter equal to 3. Finally, the performance of the model on the test dataset is assessed.

4.5.3 Hyperparameter Tuning

We execute a grid search to optimise the CNN model's hyperparameters, such as the number of filters, kernel size, hidden dimensions, and batch size. The hyperparameters that were optimised include:

- **num_filters:** Number of filters in the 1D convolutional layer (64, 128, 256).
- **kernel_size:** The convolutional kernel size (3, 5, 7).
- **hidden_dims:** Number of hidden units in the dense hidden layer (128, 250, 512).
- **batch_size:** Number of samples per gradient update (16, 32, 64).

We perform a grid search with the models' architecture and the provided hyperparameter values using the `KerasClassifier` wrapper. The models' performance on a validation set directs our choice of optimal hyperparameter values. Lastly, we evaluate the optimised model version using the test data and compare it with our baseline model.

4.6 BERT for Text Classification

4.6.1 Model Fundamentals and Design

BERT is a state-of-the-art transformer-based model renowned for its outstanding performance in various natural language processing tasks, such as text classification. In our endeavor, we fine-tune BERT to classify news articles into groups.

4.6.2 Implementation Details and Pipeline

The text data is preprocessed according to BERT's input requirements, such as tokenisation, attention masks, and input length. Then, the pre-trained BERT model is fine-tuned by adding a classification layer and training the model on the 20newsgroup dataset for three epochs.

4.6.3 Hyperparameter Tuning

The Optuna library was employed to optimise the hyperparameters of the fine-tuned BERT model. As a result, the following hyperparameters were optimised:

- Learning rate
- Number of training epochs
- Per-device training batch size

We created an objective function for the Optuna study, which trains the model using the specified hyperparameters and returns the evaluation loss. The study aimed to minimise this evaluation loss through several trials with various hyperparameter values. The models' performance on a validation set chose the optimal hyperparameter values. Finally, an optimised model is obtained using the optimal hyperparameters.

5 Experimental Setting

This section describes the precise specifications and configurations employed during the experiments.

5.1 Data set

We use the newsgroup dataset with 20 classes and 18,846 samples containing text from a discussion forum. The bydate version is utilized for performance evaluation.

5.2 Preprocessing

The dataset undergoes lowercasing, tokenisation, stopword removal, lemmatisation, and padding during preprocessing. For the Multinomial Naive Bayes model, features are extracted using TF-IDF. Word embeddings for the LSTM, CNN, and BERT models are used, with all three requiring padded input sequences.

5.3 Data Splitting

The data set, as mentioned, is divided into 60% training and 40% test sets, ensuring a balanced distribution of classes across splits. For the CNN and LSTM models, an additional validation set of 10% of the training data is used in the training process.

5.4 Model Architectures and Implementation

Four models are implemented: Multinomial Naive Bayes with TF-IDF, LSTM, CNN, and BERT with word embeddings. Each model is customised to the dataset and task at hand. Early Stopping with a 'patience' of 3 was used to prevent overfitting.

During the initial experimentation, we considered two GloVe word embeddings for the LSTM model:

1. Wikipedia 2014 + Gigaword 5 (6B tokens with 100d, 200d, and 300d vectors)
2. Common Crawl (840B tokens with 300d vectors)

Based on the performance of the LSTM model using the 300d embeddings, we decided to use the GloVe 840b (300d) embeddings for both the LSTM and CNN models in our final evaluation.

5.5 Hyperparameter Tuning

Grid search optimises Multinomial Naive Bayes, LSTM, and CNN models' hyperparameters. We use Optuna for hyperparameter optimisation in the BERT model. The optimised models performed better than their non-optimised counterparts except for the BERT model.

5.6 Evaluation Metrics

We evaluate model performance with the overall key metrics accuracy, macro average and weighted average. These metrics were selected because they provide an essential representation of each model's performance in handling the multi-class text classification task and can potentially explain imbalance issues. Additionally, we used a confusion matrix (available in the code notebook) to evaluate how well the models differentiate between each class.

6 Results and Analysis

This section presents the key takeaways from our experiments, focusing on the performance metrics and analysis of the four models (Multinomial Naive Bayes, LSTM, CNN, and BERT) on the 20newsgroup dataset.

6.1 Key Performance Metrics:

Table 1 shows the key performance metrics for each of the four models. The results are as follows:

Table 1: Key Performance Metrics

Model	Accuracy	Macro Avg F1-Score	Weighted Avg F1-Score
Multinomial	0.70	0.68	0.69
LSTM	0.64	0.63	0.64
CNN	0.67	0.66	0.67
BERT	0.70	0.69	0.70

6.2 Analysis

BERT achieves the highest performance due to its advanced pre-training and ability to capture complex contextual information. However, it requires more computational resources and training time.

CNN and LSTM models show promising results, with CNN slightly outperforming LSTM. The CNN's ability to capture local features in text data contributes to its performance, especially when handling the informal and noisy language in the dataset.

The Multinomial Naive Bayes model achieves a relatively high accuracy and serves as a strong baseline among the models. Its use of the TF-IDF feature extraction method helps it focus on the most informative words for each class, enabling it to perform well in classification despite its simplicity. Additionally, it is simple, interpretable, and computationally efficient compared to the other models.

Insights from the confusion matrix reveal that all models struggle with distinguishing between closely related classes or classes with overlapping topics. The informal language and chaotic text present in the dataset affect some models more than others, such as the Multinomial Naive Bayes model, which relies on word frequencies.

In summary, BERT performs best on the multi-class text classification task, but trade-offs in computational resources and training time must be considered when selecting a suitable model. Further improvements can be achieved through additional fine-tuning, data augmentation, or incorporating domain-specific knowledge.

7 Conclusion and future work

In conclusion, this study evaluated the performance of four different models - Multinomial Naive Bayes, LSTM, CNN, and BERT - for multi-class text classification on the 20newsgroups dataset. BERT demonstrated the best performance, owing to its advanced pre-training and ability to capture complex contextual information. However, it also required more computational resources and training time. The CNN and LSTM models delivered promising results, with CNN slightly outperforming LSTM, likely due to its capacity to capture local features in the text data. The Multinomial Naive Bayes model provided a strong baseline: simple, interpretable, and computationally efficient. However, all models faced challenges distinguishing between closely related classes or those with overlapping topics. Future improvements could involve additional fine-tuning, data augmentation, or incorporating domain-specific knowledge. Choosing the most suitable model should consider the trade-offs between performance, computational resources, and training time based on the specific application requirements.

References

[1] MonkeyLearn. "What is text classification?" (2023), [Online]. Available: `https://monkeylearn.com/what-is-text-classification/`.

[2] MonkeyLearn. "Text classification: What it is and why it matters." (2023), [Online]. Available: `https://monkeylearn.com/text-classification/`.

[3] I. H. Sarker, "Machine learning: Algorithms, real-world applications and research directions," *SN Computer Science*, vol. 2, no. 3, p. 160, 2021, ISSN: 2661-8907. DOI: `10.1007/s42979-021-00592-x`. [Online]. Available: `https://doi.org/10.1007/s42979-021-00592-x`.

[4] A. Barua, O. Sharif, and M. M. Hoque, "Multi-class sports news categorization using machine learning techniques: Resource creation and evaluation," *Procedia Computer Science*, vol. 193, pp. 112–121, 2021. DOI: `https://doi.org/10.1016/j.procs.2021.11.002`. [Online]. Available: `https://www.sciencedirect.com/science/article/pii/S1877050921021268`.

[5] S. Daud, M. Ullah, A. Rehman, T. Saba, R. Damaševičius, and A. Sattar, "Topic classification of online news articles using optimized machine learning models," *Computers*, vol. 12, 2023. DOI: `https://doi.org/10.3390/computers12010016`. [Online]. Available: `https://www.mdpi.com/2073-431X/12/1/16#`.

[6] R. Sunil. "Text categorization with support vector machines: Learning with many relevant features." (), [Online]. Available: `https://www.cs.cornell.edu/people/tj/publications/joachims_98a.pdf`.

[7] Y. Li, X. Wang, and P. Xu, "Chinese text classification model based on deep learning," *Future Internet*, vol. 10, 2018. DOI: `https://doi.org/10.3390/fi10110113`. [Online]. Available: `https://www.mdpi.com/1999-5903/10/11/113`.

[8] R. Sunil. "Naive bayes classifier explained: Applications and practice problems of naive bayes classifier." (2023), [Online]. Available: `https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/#What_Is_the_Naive_Bayes_Algorithm?`.

[9] G. Chen, D. Ye, Z. Xing, J. Chen, and E. Cambria, "Ensemble application of convolutional and recurrent neural networks for multi-label text categorization," in *2017 International Joint Conference on Neural Networks (IJCNN)*, 2017, pp. 2377–2383. DOI: `10.1109/IJCNN.2017.7966144`.

[10] Y. HaCohen-Kerner, D. Miller, and Y. Yigal, "The influence of preprocessing on text classification using a bag-of-words representation," *PLoS One*, vol. 15, 2020. DOI: `10.1371/journal.pone.0232525`. [Online]. Available: `https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7194364/`.

[11] Wikipedia, "Stop word," 2023. [Online]. Available: `https://en.wikipedia.org/wiki/Stop_word`.

[12] S. S, "Nlp essentials: Removing stopwords and performing text normalization using nltk and spacy in python," *Analytics Vidhya*, 2020.

[13] Saumyab271, "Stemming vs lemmatization in nlp: Must-know differences," *Analytics Vidhya*, 2022.