Migration Letters

Volume: 21, No: S6 (2024), pp. 262-277

ISSN: 1741-8984 (Print) ISSN: 1741-8992 (Online)
www.migrationletters.com

Enhanced Multi-Class Newsgroup Document Classification Using N-Gram Approach

Ali Abbas*[1], Manish Jaiswal[2], Shreya Agarwal[3], Prajna Jha[4], and Tanveer J.Siddiqui[5]

Abstract:
With the advancement of technology, consumption of data and information over web is increasing day by day. This has led to rising dependence of our lives on internet data at personal and professional fronts. Being a populous nation with high internet penetration, India is a huge market in digital media. The adaptability to online news, has created a large number of text news data which requires classification into categories for further applications. To address the work for efficient news document classification in the field of computational linguistics, we have proposed a novel approach for Newsgroup classification by combining tokens with a context window size of one, two and three known as unigram, bigram and trigram respectively.

In our research work, we explored the relevance of trigram by experimenting over different collection of n-grams using three supervised classifiers namely Perceptron, Nearest Centroid, and Random Forest. In the experiment, we found that inclusion of trigram along with unigram and bigram achieves the best result in terms of F1-macro score and accuracy among all three classifiers. Perceptron shows the best results in comparison with Random Forest and Perceptron classifiers, with an accuracy of 0.846945 and an F1-macro scoreof 0.83055. Significant time was spent on the validation of our efficient approach with the existing literature.

Keywords: *N-grams, Supervised Classification, Machine Learning, Artificial Intelligence, Natural Language Processing.*

1. Introduction

Today's world is not the same as it was before COVID-19. After this pandemic, we have witnessed how the internet can be used for most of our daily routines like shopping, learning, entertainment, news, and even education. India with a population of 1,407,563,842 has internet users 824,890,000 which make India thesecond country with the most internet users after China, as in (TITSPI, 2022). India is also the second largest country that produces online news content. With this much news data, we encounter the need for its classification. To overcome this problem of news data classification, we have proposed three dif[1]ferent collections of feature of N-grams token for the classification task. Text documentclassification entails the concept of pattern recognition which states that the classification of an unknown document to apredefined class with the help of the existing features of the document is document classification (Torkkola, 2004). Document classification is a sub-area spanning Natural Language Processing, Machine Learning, and Pattern Recognition (Kamath at el., 2018). In document classification, feature selection is

[1,2,3,4,5] Department of Electronics and Communication, University of Allahabad, Prayagraj,India
[1]0000-0002-7914-9574,
[2]0000-0003-2639-3984,
[3]0000-0001-8713-2679,
[4]0000-0002-2937-2487,
[5]0000-0002-8190-0646,

one of the most important task. There are many different ways of doing it, which can improve the accurate classification of a document. A collection of feature that helps to classify a document in less time is advantageous, but sometimes it becomes necessary to produce a more accurate result than the one in less time. In our experiment, we made different sets of N- gram tokens for feature selection, for classification we used three different classifiers namely Perceptron (Linear perceptron classifier), Nearest Centroid, and Random Forest Classifiers. For performance evaluation, we have used F1-macro score and Accuracy. To conduct this experiment over a news dataset we have used five classes "alt.atheism", "soc.religion.christian","talk.religion.misc", "comp.graphics", and "sci.space" of 20 Newsgroups dataset.

The primary contributions of our work are as follows:

- **Novel Methodology**: We have experimented with three different sets of N-grams with three supervised classifiers, to find the importance of each n-gram token among each other. This methodology encompasses a unique combination of n-gram features containing tri-gram tailored to the text classification problem.
- **Performance Enhancement**: Among the selected features, uni and bi -grams results significantly improve the performance metrics. This improvement includes increased accuracy, and efficiency compared to existing methods.
- **Experimental Validation**: We validate our proposed approach through rigorous experimentation and analysis. A total of nine experiments have been conducted to find the best feature set of n-gram.
- **Comparative Analysis**: We performed comprehensive comparative analysis with existing literature. This analysis highlights the strengths and weaknesses of our approach and provides insights into its competitiveness.
- **Open-Source Implementation:** Our work is performed in an open source programing language Python, all the library used in this experiment are publicly available. The dataset is also an open source dataset and is also associated directly with the python library.

Overall, our work makes significant contributions in finding the best feature set of n-gram for text classification, with extensive experimentation and dataset insights.

Rest of this paper is structured as follows. In section 2, related work in the proposed area has been discussed. Section 3, includes background concepts and proposed methodology of the work. In section 4, experimental results and performance evaluation is described. Furthermore comparative study of our work with existing work has been presented. Finally in section 5, conclusion and future direction of the work is discussed.

2. Related Work

Classification of news articles and news data is now a trending research area (Dilrukshi at el., 2013; Tariq at el., 2022; Patil at el., 2022; Zhao at el, 2022). Even the classification of news in other foreign languages is a trending topic (El Rifai at el, 2022; Barud & Salau, 2022). For the classification of text news documents, we first have to do the feature selection. There are plenty of state-of-the-art feature selection methods like bags of words, tf-idf, etc., which give promising results. They somehow give the result but unfortunately, they lose the positional information of the tokens or words, which affects their accuracy. To overcome this situation many have come up with many solutions, some of them were effective and some of them were not. N-gram based text classification was introduced by William B. Cavnar and John M. Trenkle for creating n-grams of letters which was used for the correction of spelling and finding similar words (Cavnar & Trenkle, 1994).

In (Yan al el., 2023), author introduces ANT-STR, an adaptive n-gram transformer tailored for multi-scale scene text recognition, addressing the dearth of transformer-based solutions

in this domain. ANT-STR utilizes an adaptive n-gram embedding to optimize patch sizes for semantic correlation exploration, alongside a patch-based n-gram attention mechanism to enhance feature extraction from multi-scale texts. Experimental evaluations on benchmark datasets, including a new Indonesian tourism scene text dataset, showcase ANT-STR's significant advancements over existing methods, particularly in handling complex multi-scale scene texts.

In (Xiang, 2024), discusses the application of natural language processing (NLP) in classifying portions of bone marrow report text into their appropriate sections. Using a classical NLP algorithm involving n-grams and K-means clustering, 1480 bone marrow reports were processed. The study evaluated various parameters such as token replacement, n-grams, and the number of centroids, identifying an optimal NLP model with an ensemble algorithm. The reported result achieved 89% accuracy and demonstrated the effectiveness of classical NLP approaches in accurately categorizing medical text.

Fürnkranz (1998) proposed an algorithm for creating N-grams of words or N-grams of tokens with limited frequency provided by the users. His method was effective till the time n-grams are selected with Tf-Idf, but was producing bad results when the selection of n-grams was on the basis of high document frequency.

Researches (Shah at el, 2020) compared classifier models like Logistic Regression, Random Forest, and KNN Models to obtain better results in non – linear classifiers; Multilayer Perceptron Classifier used with Support Vector Machine is evaluated with good results with non-linear classification (Suykens, & Vandewalle, 1999).

We extended our study with different sets of N-grams on different classifiers, without leaving the features on a frequency basis. To perform this we have chosen Perceptron, Nearest Centroid, and Random Forest Classifiers.

In (Tellez at el., 2018), author developed text classifier for multi-propose use which can handle tasks even without a great knowledge of related domain. For performance evaluation, proposed work was compared on 30 datasets and the performance of the model was better.

Conway at el., (2009) uses n-grams and semantic features technique for classifying BioCaster disease outbreak reports. Use of semantic tagger for feature generation was novelty of the proposed work. There were mainly three features used based on named Entity, n-gram, and features explored from USAS semantic tagger. Author used three machine learning classifier in work Support Vector Machine, anddecision tree algorithm and Naïve Bayes algorithm. The highest accuracy was achieved in case of Naïve Bayes algorithm with conjunction of features derived from USAS semantic tagger.

Feature selection is highly useful in reduction of dimension of data and it is very important concept in document classification. In (Tang at el., 2019), author employs 5- dimensional joint mutual information to calculate interaction terms. Generally, 2-3 dimensional information is not too much effective for handling high order interaction. To avoiding features overestimation, non-linear approach has been used in the work.

In (Lertnattee, & Theeramunkong, 2004), author used three types of term distributions namely inter-class, intra-classand in-collection distributions with term frequency (tf) and inverse document frequency (idf) to improving centroid based text categorization. Different types of centroid based classifiers were used in the proposed work and performance was done by comparing proposed classifier with KNN and Naïve Bayes classifiers algorithm.

In existing literature, we noted that classification challenges were typically tackled using computationally demanding algorithms or feature extraction methods that require substantial computational power. Furthermore, while extensive research has been conducted on n-grams, the collective analysis of feature sets derived from different n-gram combinations has been lacking. To address these gaps, we've devised three sets of n-grams: uni-gram, uni and bi-gram, and uni, bi, and tri-gram. In this study, we are assessing the efficacy of these sets using three distinct supervised classifiers—Perceptron, Nearest Centroid, and Random Forest—in comparison to prior research efforts. Our aim is to

discern any notable enhancements in classification performance.

3. Classifiers Description

Perceptron (Linear perceptron classifier)

The perceptron is a linear classifier that separates classes using a decision boundary by its learning in feature space. It is a neural network model consisting of a single neuron or node. The input of this single neuron is a single row of data for the prediction of the class label. To predict the class label, the perceptron computes the weighted inputs and sums it with the bias (default set to 1). This biased weighted sum of input is known as activation.

$$\text{Activation}\,(z) = \text{Input} * \text{Weight} + \text{Bias} \ldots\ldots\ldots\ldots\ldots \text{(i)}$$

The linear perceptron classifier is a simple binary classification algorithm that learns a linear decision boundary to separate data points of different classes. It takes input features, assigns weights to them, and produces an output based on a threshold function. During training, the weights are adjusted iteratively to minimize classification errors, making it a foundational concept in machine learning and the precursor to more advanced neural network architectures. The formula for a linear perceptron classifier is represented by the following equation:

$$y = \text{sign}(z) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(ii)}$$

Where y is the output of the perceptron (predicted class), sign is the sign function, which outputs +1 if the argument is positive, -1 if it's negative, and 0 if it's zero.

Nearest Centroid

A nearest centroid is a type of classifier similar to KNN classifier. The basic classification technique used in nearest centroid is to compute the centroid of each target class and measuring the distance between the input point and the centroid if each class. The minimum distance measured will be considered as the class of that particular point.
The mathematical equation used by the nearest centroid classifier is relatively simple. Given a set of feature vectors $X = \{x_1, x_2, \ldots\ldots, x_n\}$ where each x_i is a feature vector, and a set of centroids $C = \{c_1, c2, \ldots\ldots, c_k\}$ where c_j is the centroid of a class j, the prediction for a new feature vector x is made by finding the closest centroid to x using some distance metric d(x, c). Here is the basic mathematical representation:

$$c_j = \frac{1}{|S_j|}\sum_{x_i \in S_j} x_i \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(iii)}$$

Where c_j is the centroid of the class j, and S_j is the set of feature vector belonging to class j.
The distance between a feature vector x and a centroid c_j is calculated using the Euclidean distance $d(x, c_j)$.

$$d(x, c_j) = \sqrt{\sum_{i=1}^{n}(x_i - c_{ji})^2} \ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(iv)}$$

Where n is the number of features and c_{ji} is the i^{th} component of the centroid c_j.
Given a feature vector x, the predicted class label y is the class whose centroid is closest to

x:

$$y = \arg_j \min d(x, c_j) \ldots\ldots\ldots\ldots\ldots\ldots\ldots\ldots \text{(v)}$$

In other words, y is assigned the class label corresponding to the centroid c_j that minimizes the distance $d(x, c_j)$.
So, the prediction formula essentially finds the centroid c_j that is closest to the given feature vector x, based on the distances calculated using a chosen distance metric, and assigns the class label associated with that centroid to the feature vector x.

Random Forest

Random forest is a supervised machine learning algorithm proposed by L. Breiman and A. Cutler used for both classification and regression approaches. The basic functioning approach of the random forest algorithm is to average the result of multiple subsets of decision trees to produce more accurate results in prediction.

Training Phase:

- We create many decision trees, each based on a different subset of the training data.
- Each tree is trained to make decisions based on a random selection of features.
- This randomness helps each tree learn different aspects of the data.

Prediction Phase:

- When we want to predict the class of a new sample:
- We ask each tree in the forest to predict the class.
- Then, we look at all the predictions from all the trees.
- The final prediction is the one that the majority of the trees agree upon.

For each new input sample x, let $T_i(x)$ represent the prediction made by the i^{th} decision tree. The final prediction $\hat{y}$ is determined by majority voting:

$$\hat{y} = \text{mode}\{T_1(x), T_2(x), \ldots\ldots, T_n(x)\} \ldots\ldots\ldots\ldots\ldots \text{(vi)}$$

Here, n is the total number of decision trees in the forest. We consider the prediction that occurs most frequently among all the decision trees for the input sample x as the final prediction$\hat{y}$.

4. Proposed Methodology

In this section, we delve into the workings of our proposed model. Initially, we address the pre-processing steps applied to the dataset, followed by the feature selection process. Subsequently, we transition to model training, wherein we outline the parameters selected for the classifiers. Finally, we elaborate on the testing procedures implemented for our model and the evaluation method employed.

Pre-processing

In feature extraction, we are removing the English stopwords present in our dataset. Stopwords are those words that frequently occur in the dataset and do not provide any useful information to differentiate between classes.
After the pre-processing, we perform the tokenization over the remaining words present in

the documents.

Feature Extraction

We have implemented a filtering mechanism to remove tokens with a length equal to or less than two characters. The rationale behind this decision is rooted in the sheer size of the corpus, where tokens of such short length tend to exhibit either sparse or dense distributions, thereby introducing noise into the dataset.
For our experimental setup, we are leveraging N-grams as the primary features. Specifically, we have selected unigrams, bigrams, and trigrams as our N-gram types. The term "N-grams" refers to sequences of N words occurring within a specified window. In this context, "uni," "bi," and "tri" signify one, two, and three words respectively, encapsulated within a window. Consequently, a unigram encompasses a single word within the window, a bigram comprises two words, and a trigram encompasses three words.

Example:- Consider the following sentence:
"The Sun rises in the East"

- Unigram ['The", "Sun", "rises", "in", "the", "East"]
- Bigram ["The Sun", "Sun rises", "rises in", "in the", "the East"]
- Trigram ["The Sun rises", "Sun rises in", "rises in the", "in the East"]

In this experiment, we employ three distinct types of N-gram combinations: 'unigram', 'unigram and bigram', and 'unigram, bigram, and trigram'.
Following the generation of our Feature Collections, the dataset is partitioned into training and testing subsets. Beginning with our training dataset, we transform tokens into vectors by capturing their frequencies across all documents.

$$t_i = t_{i1}, t_{i2}, \dots\dots, t_{ij} \quad \text{(vii)}$$

Where t_i is the token at i^{th} index containing $t_{i1}, t_{i2}, \dots\dots, t_{ij}$ frequency term t_i in allthe documents j.

Model Training

After extracting the features from the dataset, the next step involves partitioning the dataset into two distinct subsets: the training dataset and the testing dataset. The training dataset is utilized to train our classifier, while the testing dataset is employed to evaluate the performance of the classifier. For classification tasks, we have selected three state-of-the-art classifiers. The training process workflow is depicted in Figure 1.

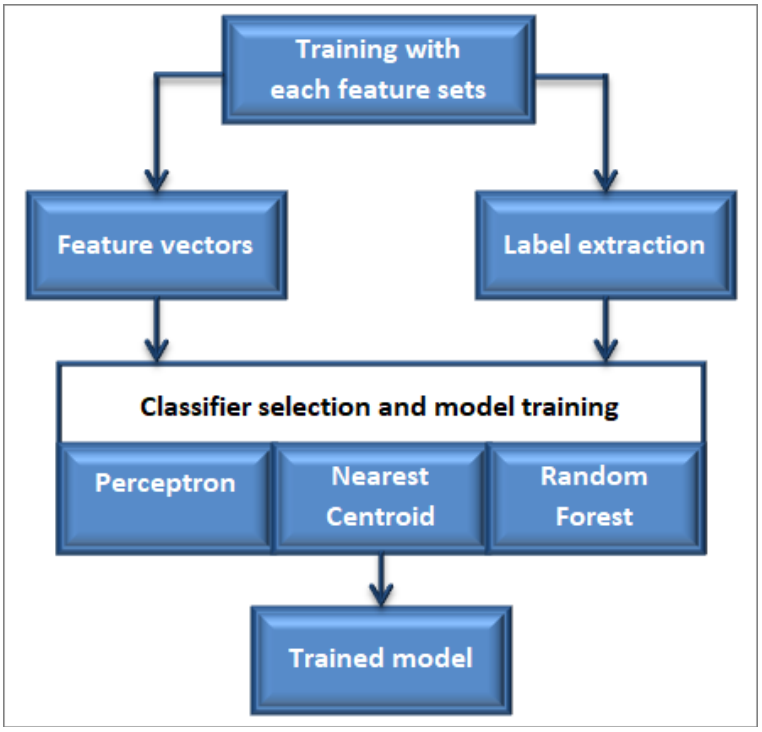

Figure 1. Training Classification Model

After preparing the feature vectors for all sets, a separate list containing their corresponding labels or classes is generated. These feature vectors, along with their labels, are sequentially passed to each of the three classifiers to obtain trained models. The training process is outlined in Figure 1, resulting in the generation of a total of nine distinct models.

Classifiers Parameter Setting

Following are the parameter setting of classifier used in this experiment. We have used sklearn library for all classifiers used in this experiment.

Random Forest Classifier

The Random Forest Classifier constructs a forest of decision trees, with each tree in the forest being trained on a random subset of the training data. The number of trees in the forest, controlled by the `n_estimators` parameter, is set to 100. During the construction of each tree, the algorithm uses the 'gini' criterion to measure the impurity of a split.
The Gini impurity is calculated as follows:

$$\text{Gini}(D) = 1 - \sum_{j=1}^{c}(p_j)^2 \quad \text{..... (viii)}$$

Where: D is the dataset at a particular node.
c is the number of classes.
P_j is the proportion of instances of class j in the node.

The maximum number of features considered for splitting nodes, governed by the `max_features` parameter, which selects the square root of the total number of features. The depth of the trees, controlled by the `max_depth` parameter, is unconstrained, allowing nodes to expand until they are pure or contain fewer samples than required for splitting.

Additionally, the settings specify that a node must contain at least two samples (`min_samples_split`) to be considered for splitting and that each leaf node must have at least one sample (`min_samples_leaf`). Finally, the Random Forest Classifier performs bootstrap sampling, meaning that each tree is trained on a bootstrapped sample of the training data. These settings are used for building and training Random Forest models.

Nearest Centroid Classifier

The Nearest Centroid Classifier is using Euclidean distance metric to calculate the distance between data points and centroids. The centroids represent the mean coordinates of the samples for each class in the feature space. During classification, the algorithm assigns a new data point to the class with the nearest centroid based on the calculated distances. The shrinkage factor, controlled by the 'shrink_threshold' parameter, is set to none, meaning that no shrinkage is applied, which means the centroids will not shrink towards the overall mean of the dataset.

Perceptron

The Perceptron utilizes gradient descent for training, a process that involves iteratively updating the model's weights based on individual training samples. The classifier setting for the regularization parameter (`alpha`) is 0.0001, controlling the strength of regularization when using 'l2' penalties. The fit intercept parameter is set to True, indicating that an intercept term is included in the decision function. The maximum number of iterations (`max_iter`) is set to 1000, determining the maximum number of iterations for the optimization algorithm to converge. The tolerance (`tol`) is set to 1e-3, specifying the stopping criterion based on the change in the loss function. The initial learning rate (`eta0`) is set to 1.0, and the learning rate schedule (`learning_rate`) is set to 'constant'.

Model Testing

After the training process, we have our trained classifier models. With this model, we are further going to test our model using testing dataset. In this testing process first, we fetch the feature vector, then send it to the predicted class and then compare it with its original class/label, explained in figure 2. The results are evaluated using confusion matrix.

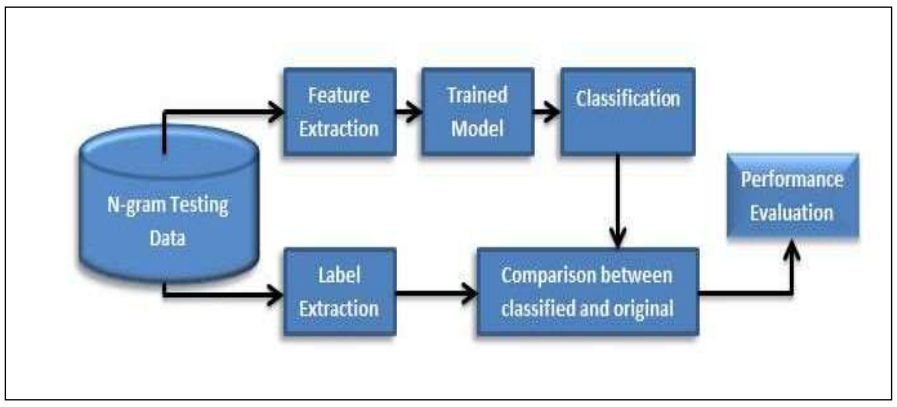

Figure. 2 Testing Classification Model

Evaluation Metrics

In assessing our results, we employed several performance measures including the confusion matrix, precision, recall, macro F1-score, and accuracy.

Confusion Matrix: A tabular representation showcasing the predicted outcomes of a classification problem constitutes the confusion matrix.
Precision: Precision denotes the fraction of retrieved documents that are pertinent to the query. It is computed as the ratio of True Positives to the sum of True Positives and False Positives.

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive + False Positive}} \text{ ..... (ix)}$$

Recall: Recall signifies the fraction of relevant documents successfully retrieved. It is calculated as the ratio of True Positives to the sum of True Positives and False Negatives.

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive + False Negative}} \text{ ..... (x)}$$

Macro F1-score: The macro average of the harmonic mean of recall and precision is referred to as the Macro F1-score. It provides a balanced assessment of the model's precision and recall across all classes.

$$\text{Macro F1} = 2 * \frac{|\text{Precision} * \text{Recall}|}{\text{Precision+Recall}} \text{ ..... (xi)}$$

Accuracy: The accuracy of our method is determined using the binary classification accuracy formula. It calculates the proportion of accurately predicted outcomes among the total number of samples. The formula sums the correct predictions and divides them by the total number of samples evaluated.

$$\text{Accuracy} = \frac{1}{n_{samples}} \sum_{i=0}^{n_{samples}-1} (\hat{y}_i = y_i) \text{ ..... (xii)}$$

Here, if $\hat{y}_i$ denotes the predicted value for the i^{th} sample, and y_i represents the actual true value, then the proportion of accurate predictions among $n_{samples}$ is determined as per equation (xii).

5. Experiment and Result Analysis

This section includes the dataset used in this work, experimental results, performance evaluation and comparative analysis with other's work on the same dataset.

Dataset

The 20 Newsgroups dataset comprises 20 distinct news groups, encompassing roughly 20,000 news documents. In our experiment, we focused on a subset of this dataset by selecting five classes out of the 20 available. The selection of these five classes was based on their diversity, as some classes within the dataset exhibit significant similarity to one another. By choosing classes that offer a wide range of topics and content, we aim to enhance the variety and representativeness of the data used in our experiment as shown in table I.

Table I. Dataset Description.

S. No.	Selected Class			
	Class	Total Doc.	Training	Testing

1	alt.atheism	799	480	319
2	soc.religion.christian	997	599	398
3	talk.religion.misc	628	377	251
4	comp.graphics	973	584	389
5	sci.space	987	593	398

The selected five classes from the 20 Newsgroups dataset are alt.atheism, soc.religion.christian, talk.religion.misc, comp.graphics, and sci.space. Combined, these classes comprise a total of 4,384 documents. We partitioned this dataset into two subsets: a training dataset and a testing dataset. The training dataset consists of 1,751 documents, while the testing dataset comprises 2,633 documents. The size of the training dataset is approximately 5.34 MB, while the testing dataset occupies approximately 3.87 MB of storage space. This division allows us to effectively train and evaluate our classifiers on distinct sets of data, facilitating comprehensive analysis and validation of our models.

Experimental Setup

The system configuration on which this experiment was performed is as follows. Processor: Intel(R) Core(TM) i3-4030U CPU @ 1.90GHz, 1900 Mhz, 2 Core(s), 4 Logical Processor(s), Total physical memory (RAM) 7.89 GB. Python served as our programming language of choice for this project, and for classification purposes, we relied on the sklearn library.

Feature Vector Generation

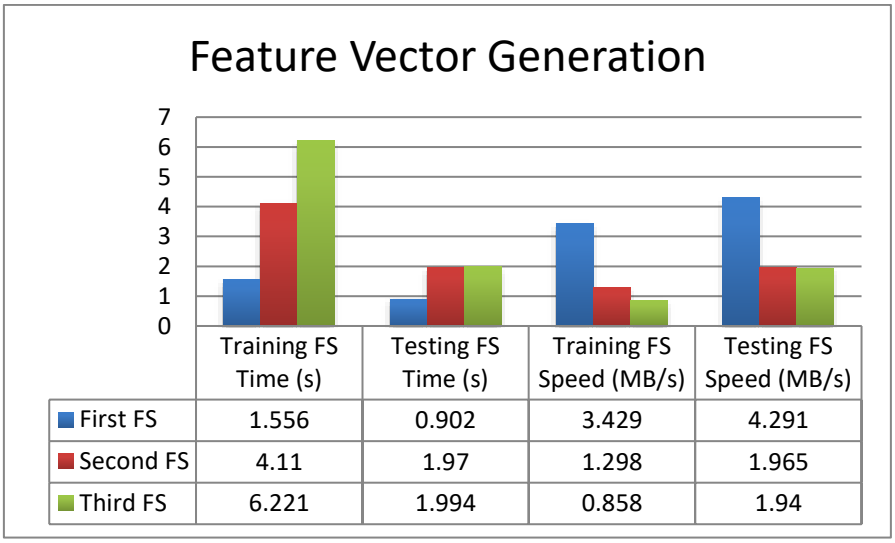

Figure 3. Time taken for feature vector generation

In our case as shown in figure 3, generation of the training feature vector has taken 1.556s at 3.429MB/s for 1st Feature Collection, 4.110s at 1.298 MB/s for 2nd Feature Collection, and 6.221s at 0.858 MB/s for 3rd Feature Collection.
The testing feature vector has taken 0.902s at 4.291 MB/s for 1st Feature collection, 1.970s at 1.965 MB/s for 2nd Feature collection, and 1.994s at 1.940 MB/s for 3rd Feature Collection.

Confusion Matrix

In assessing the classification algorithm's performance, we employed the confusion matrix as our evaluation metric. We analyzed a total of nine confusion matrices derived from various experiments. Here, we focus on evaluating the results using the confusion matrix of the Perceptron classifier, incorporating all three n-grams as features. In Figure 4, the diagonal of the matrix signifies the true positive values, providing valuable insights into the classifier's accuracy and performance.

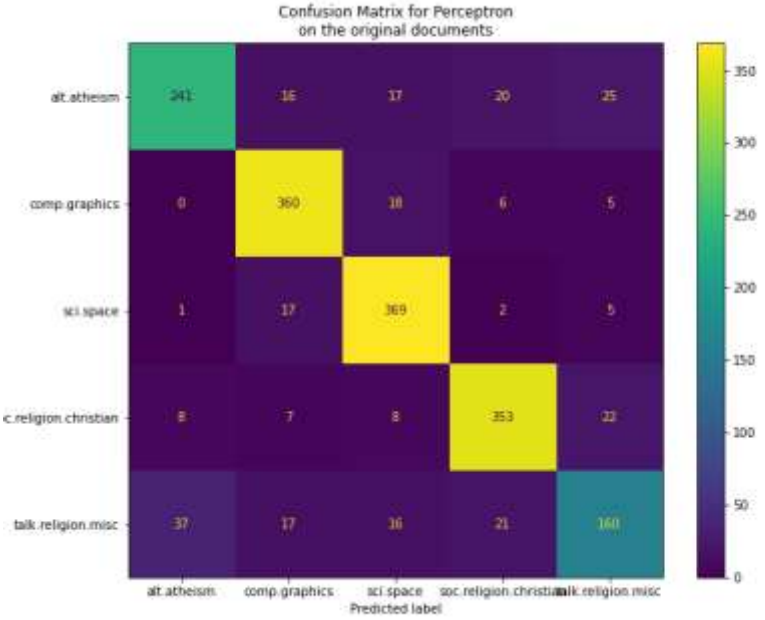

Figure 4. Confusion Matrix of Perceptron using trigram approach

A higher true positive value indicates better precision, recall, F1-score, and accuracy. Observing the confusion matrix, we note that the majority of documents are accurately classified into their respective classes. This suggests that the classifier performs well in identifying documents correctly, thereby contributing to improved precision, recall, F1-score, and accuracy metrics.

Precision and Recall

As anticipated from the predictions in the confusion matrix, the precision, recall, and F1-score values exhibit notable performance, as indicated in Table II. This further confirms the classifier's effectiveness in accurately classifying documents and underscores its robustness in achieving high precision, recall, and F1-score metrics.

Table II. Performance Matrix of Perceptron over trigram

Classes	**Precision**	**Recall**	**F1-score**
alt.atheism	0.839721	0.755486	0.79538
comp.graphics	0.863309	0.92545	0.8933
sci.space	0.86215	0.936548	0.89781
soc.religion.christian	0.878109	0.886935	0.8825
talk.religion.misc	0.737327	0.63745	0.683761

Macro Average	**0.836123**	**0.828374**	**0.83055**

First Feature Collection (Unigram)

For the 1st Feature Collection, our dataset comprises 2,633 samples for training and 1,751 samples for testing. The total number of features in the first feature set amounts to 38,614 features. Table III presents the outcomes achieved across the three classifiers, providing a comprehensive overview of their performance on the dataset.

Table III 1st Feature Collection

S. No.	**Performance Evaluation**		
	Classification Algorithm	**F1-macro**	**Accuracy**
1	Perceptron (Linear perceptron classifier)	0.82091	0.838378
2	Random Forest	0.793794	0.818961
3	Nearest Centroid	0.622598	0.637350

Second Feature Collection (Unigram and bigram)

In the second feature collection, we have 2633 samples for training and 1751 samples for testing. The total number of features in second feature set is 312795 features Table IV contains the results obtained over the three classifiers.

Table IV 2nd Feature Collection.

S. No.	**Performance Evaluation**		
	Classification Algorithm	**F1-macro**	**Accuracy**
1	Perceptron (Linear perceptron classifier)	0.833584	0.850942
2	Random Forest	0.861849	0.786979
3	Nearest Centroid	0.646136	0.659623

Third Feature Collection (Unigram, bigram, and trigram)

Within the third feature collection, we employed 2,633 samples for training and 1,751 samples for testing. The total number of features in the third feature set amounts to 645,786 features. Table IV presents the results obtained across the three classifiers, offering insights into their performance on the dataset.

Table V 3rd Feature Collection.

S. No.	**Performance Evaluation**		
	Classification Algorithm	**F1-macro**	**Accuracy**

1	Perceptron (Linear perceptron classifier)	0.83055	0.846945
2	Random Forest	0.750626	0.772701
3	Nearest Centroid	0.657120	0.669331

Discussion

The performance evaluation across all three feature collections has distinctly revealed that the highest overall accuracy and F1-macro score are attained in the second collection of features, which encompasses the combination of unigram and bigram. Notably, the feature vector generation time for all three n-grams is longest, while the first feature set exhibits the best timing performance. Conversely, the second feature set demonstrates average timing, falling between the extremes of the first and third sets. Among these classifiers, the Perceptron model emerges as the top performer within this feature collection, boasting an accuracy of 0.850942 and an F1-macro score of 0.833584. This marks the highest achievement across all three feature collections and classifiers evaluated.

Performance comparisons with existing literature

The comparison of our best result is done with the existing literature on the same dataset is shown in Table VI.

Table VI Comparison Matrix of existing literature

S. No.	**Method**	**Ref. No.**	**Result in % age**	**Original Result**
1	KNN	(Jiang at el, 2018)	61.68	61.68
2	SVM		81.60	81.60
3	DBN + Softmax(1)		65.33	65.33
4	DBN + Softmax(2)		82.63	82.63
5	Linear SVM	(Defferrard at el., 2016)	65.90	65.90
6	Multinomial Naive Bayes		68.51	68.51
7	Softmax		66.28	66.28
8	FC2500		64.64	64.64
9	FC2500-FC500		65.76	65.76
10	GC32		68.26	68.26
11	MI-Kernel	(Zang at el., 2024)	51.5	0.515
12	MILES		53.9	0.539
13	mi-SVM		64.1	0.641
14	mi-SPSVM		47.5	0.475

15	DSMIL-T		68.0	0.68
16	DSMIL-H		70.7	0.707
17	CNB +TF	(Abbas at el., 2023)	82.11	0.8211
18	**rceptron +Uni and bigram. oposed)**		**85. 09**	**0.850942**

Figure 5. Performance comparison graph

Table VI presents a comparative analysis of our proposed method with sixteen alternative approaches as delineated in references (Jiang at el, 2018; Defferrard at el., 2016; Abbas et al., 2023; Zhang at el., 2024). Representation of performance comparison in graph is presented in Figure 5. Notably, the accuracy of our proposed model surpasses that of all sixteen methods. While reference (Jiang at el, 2018) achieved a slightly higher accuracy than our model utilizing a hybrid DBN + Softmax(2) architecture, the complexity of their model architecture (2000-1000-500-20) imposes significant resource and time requirements. The marginal increase in accuracy is negligible when considering overall performance factors such as accuracy, time complexity, resource usage, and space complexity. Defferrard at el., (2016) introduced CNNs within the context of spectral graph theory, offering the same linear computational complexity and constant learning complexity as classical CNNs, while being applicable to any graph structure. However, their performance falls short compared to our proposed model. Abbas et al., (2023) explored various Naive Bayes variants to identify the best variant using different weighting techniques. The performance of the best Naive Bayes variant yielded lower accuracy and F1-score. Additionally, reference (Zhang at el., 2024) employed a novel Multi-instance learning approach with a double similarities weighted multi-instance learning kernel framework, achieving a maximum accuracy of 70.7 percent on the 20 Newsgroups dataset, nearly 10 percent lower than our proposed model.

6. Conclusions and future scope

This paper presents the implementation of three different classifiers on three feature collections comprising unigram, bigram, and trigram representations of the 20 Newsgroups dataset. The experimental findings indicate that the feature collection incorporating unigram and bigram combinations yields superior results compared to the other collections. Notably, the Perceptron classifier consistently outperforms other classifiers across all feature collections, achieving an accuracy of 0.850942 and an F1-macro score of 0.833584. These results underscore the effectiveness of utilizing a combination of unigram and bigram features with the Perceptron classifier.
Future scope includes exploring deep learning approaches for document classification and applying the proposed approach on other domains of documents.

Acknowledgement

The corresponding author wants to thank Mrs. Gulnaaz Fathima Rizvi, Miss Komal Singh, and Dr. Arati Kushwaha for their emotional support and guidance.

References:

[1] Cavnar, W. B., & Trenkle, J. M. (1994, April). N-gram-based text categorization. In Proceedings of SDAIR-94, 3rd annual symposium on document analysis and information retrieval (Vol. 161175, p. 14). https://dsacl3-2019.github.io/materials/CavnarTrenkle.pdf

[2] TITSPI (2022, July). The Indian Telecom Services Performance Indicators of Telecom Regulatory Authority of India. https://www.trai.gov.in/sites/default/files/QPIR_26072022_0.pdf

[3] Torkkola, K. (2004). Discriminative features for text document classification. Formal Pattern Analysis & Applications, 6, 301-308. https://doi.org/10.1007/s10044-003-0196-8

[4] Kamath, C. N., Bukhari, S. S., & Dengel, A. (2018, August). Comparative study between traditional machine learning and deep learning approaches for text classification. In Proceedings of the ACM Symposium on Document Engineering 2018 (pp. 1-11). https://doi.org/10.1145/3209280.3209526

[5] Dilrukshi, I., De Zoysa, K., & Caldera, A. (2013, April). Twitter news classification using SVM. In 2013 8th International Conference on Computer Science & Education (pp. 287-291). IEEE. https://doi.org/10.1109/ICCSE.2013.6553926

[6] Tariq, A., Mehmood, A., Elhadef, M., & Khan, M. U. G. (2022). Adversarial Training for Fake News Classification. IEEE Access, 10, 82706-82715 https://doi.org/10.1109/ACCESS.2022.3195030.

[7] Patil, D., Lokare, R., & Patil, S. (2022, May). An Overview of Text Representation Techniques in Text Classification using Deep Learning Models. In 2022 3rd International Conference for Emerging Technology (INCET) (pp. 1-4). IEEE. https://doi.org/10.1109/INCET54531.2022.9825389

[8] El Rifai, H., Al Qadi, L., & Elnagar, A. (2022). Arabic text classification: the need for multi-labeling systems. Neural Computing and Applications, 34(2), 1135-1159. https://doi.org/10.1007/s00521-021-06390-z

[9] Zhao, W., Zhu, L., Wang, M., Zhang, X., & Zhang, J. (2022). WTL-CNN: A news text classification method of convolutional neural network based on weighted word embedding. Connection Science, 34(1), 2291-2312. https://doi.org/10.1080/09540091.2022.2117274

[10] Barud, D. W., & Salau, A. O. (2022). Detection of fake news and hate speech for ethiopian languages: A systematic review of the approaches. Journal of Big Data, 9(1) https://doi.org/10.1186/s40537-022-00619-x

[11] Fürnkranz, J. (1998). A study using n-gram features for text categorization. Austrian Research Institute for Artifical Intelligence, 3(1998), 1-10. https://ofai.at/papers/oefai-tr-98-30.pdf

[12] Shah, K., Patel, H., Sanghvi, D., & Shah, M. (2020). A comparative analysis of logistic regression, random forest and KNN models for the text classification. Augmented Human Research, 5, 1-16. https://doi.org/10.1007/s41133-020-00032-0

[13] Suykens, J. A., & Vandewalle, J. (1999). Training multilayer perceptron classifiers based on a modified support vector method. IEEE transactions on Neural Networks, 10(4), 907-911.

https://doi.org/10.1109/72.774254

[14] Tellez, E. S., Moctezuma, D., Miranda-Jiménez, S., & Graff, M. (2018). An automated text categorization framework based on hyperparameter optimization. Knowledge-Based Systems, 149, 110-123. https://doi.org/10.1016/j.knosys.2018.03.003

[15] Conway, M., Doan, S., Kawazoe, A., & Collier, N. (2009). Classifying disease outbreak reports using n-grams and semantic features. International journal of medical informatics, 78(12), e47-e58. https://doi.org/10.1016/j.ijmedinf.2009.03.010

[16] Tang, X., Dai, Y., & Xiang, Y. (2019). Feature selection based on feature interactions with application to text categorization. Expert Systems with Applications, 120, 207-216. https://doi.org/10.1016/j.eswa.2018.11.018

[17] Lertnattee, V., & Theeramunkong, T. (2004). Effect of term distributions on centroid-based text categorization. Information Sciences, 158, 89-115. https://doi.org/10.1016/j.ins.2003.07.007

[18] Jiang, M., Liang, Y., Feng, X., Fan, X., Pei, Z., Xue, Y., & Guan, R. (2018). Text classification based on deep belief network and softmax regression. Neural Computing and Applications, 29, 61-70. https://doi.org/10.1007/s00521-016-2401-x

[19] Defferrard, M., Bresson, X., & Vandergheynst, P. (2016). Convolutional neural networks on graphs with fast localized spectral filtering. Advances in neural information processing systems, 29. https://proceedings.neurips.cc/paper_files/paper/2016

[20] Yan, X., Fang, Z., & Jin, Y. (2023). An adaptive n-gram transformer for multi-scale scene text recognition. Knowledge-Based Systems, 280, 110964. https://doi.org/10.1016/j.knosys.2023.110964

[21] Xiang, R. F. (2024). Use of n-grams and K-means clustering to classify data from free text bone marrow reports. Journal of Pathology Informatics, 15, 100358. https://doi.org/10.1016/j.jpi.2023.100358

[22] Abbas, A., Jaiswal, M., Agarwal, S., Jha, P., & Siddiqui, T. J. (2023, March). Performance Based Comparative Analysis of Naïve Bayes Variants for Text Classification. In International Conference on Data Science and Communication (pp. 295-310). Singapore: Springer Nature Singapore. https://doi.org/10.1007/978-981-99-5435-3_20

[23] Zhang, J., Wu, Y., Hao, F., Liu, X., Li, M., Zhou, D., & Zheng, W. (2024). Double similarities weighted multi-instance learning kernel and its application. Expert Systems with Applications, 238, 121900. https://doi.org/10.1016/j.eswa.2023.121900