**Transfer Learning For Classifying Research Papers**

Submitted By,

Rezoanul Islam

**ID: 2017-2-60-129**

Ruhadina Musfikin

**ID: 2017-1-60-002**

Raihan Mohammad Karmol Ruhani

**ID: 2017-2-60-143**

**Supervisor**

**Rashedul Amin Tuhin**

**Senior Lecturer**

**Department of Computer Science & Engineering**

**East West University**

Chapter 1

# Introduction

**1.1 Problem Statement**

Text classification is a process to label text from unstructured texts in order to categorize, which has been a hot topic for topic modelings in machine learning over the decades. Text classification or text categorization has been used for various purposes such as business development. In the era of science and technology field of research puts a significant as we set new parameters of science and technology, we breach through new production of technology, but to perform such study or conduct investigation in such field or in any area of topics and subjects researchers, scientists need to gather or collect research materials and study substance which is abundantly available and depended on internet and for browsers being available researchers have the ability to access anything from. A lot can be done but these technologies have their limitations which is problematic and can be time consuming issues for scientists. With that being said, let us focus on the problem of searching for required information and ending up with something in return which is not related to the field of study or a lot of other information where scientists filter out by themselves in order to have the exact information or datas. As we know, browsers and other internet options for searching purposes use mechanisms with Artificial Intelligence or filtering, and this process is not a sophisticated method which can be used for a result oriented system which makes the system of our searching period to gather useful and needed information. As the process of search may be a fully-automated look motor that employs software known as web crawlers that investigate the internet frequently to discover pages to include to our list. In truth, the endless larger part of pages recorded in our comes about are not physically submitted for inclusion, but are found and included naturally when our web crawlers investigate the internet. Various investigation papers have been distributed online as well as offline with the increasing progress of computer and data innovations, which makes it difficult for clients to look and categorize their curiously investigating papers for a specific subject [1].

We presenting a advance method to improve the process of finding research articles by multiclass classification, which is much more time efficient in a comparative study, where we utilized several Natural Language Processing (NLP) techniques such as Tokenization, lemmatization to pre process the abstract section and traine four (Long Short Term Memory, Gated Recurrent Unit, --Long Short Term Memory, Bi-directional, Bi-directional Gated Recurrent Unit) deep Learning algorithms on three different word embedding (word2vec, GloVe) techniques to perform multiclass text classification to identify scientific articles.

**1.2 Motivation**

Our study shows and gives us a process which minimizes and outlines the problem of searching issues of finding expected results for research papers and materials. This study develops a process where finding the research papers can be an issue of comfort as our recent and till now like browsers gets confused about finding specific papers for expected study. Researchers and scientists gather information in order to perform study or experiment in their related field of education where gathering information is a hustle as they get a lot of unnecessary and unrelated information and researchers have to find out manually, filtering through reading. As these tools offer extra and unwanted, unnecessary information in case finding computer science papers may suggest papers that are related with mathematics or statistics. They create a confusion as the fields of these study areas are saturated, a topic can be related to some other fields of study consequently such as a research paper maybe related and fall under both Computer Science or Statistics or Mathematics.

Till now studies on this very topic suggest some approaches which are lacking in their mechanism and methods. Some perform the process of searching with keywords where finding out papers and other materials through targeting the keywords like for computer science, deep learning other techniques but in such cases a challenges as a case where user or researchers not using the exact keywords confusion and complications may arise through the process which can be problematic, getting the expected materials or papers with wrong keywords point out null, or abit of lengthy. Issues such as these may arise if researchers mistakenly input wrong keywords and not enough suitable words.

Also, a case has risen where the process is unsupervised and not classified dataset, where the machine had train itself from the data in order to classify or categorize form datasets and with not enough data or due lack of data the process is not performing a better result [1]. Unsupervised learning is an algorithm that studies patterns of unlabeled data. It is believed that with the help of imitation, an important method of learning between people, the machine is forced to create a small internal image of its world, and then create a figurative meaning out of it.

Our study will offer our fellow researchers to sort out the papers in which segmentation they fall. To minimize the issues our study develops, automated categorization of research papers based on topic is time efficient. Prior research works have been done on this topic but their studies performed in a poor accuracy score. We worked on multiclass (Computer science , Mathematics, statistics, Physics) and obviously to outperform the results of prior research which has been on research paper classifying. In a nutshell, we approached the transfer learning process to conduct our research on this topic, where it performs better due to prior other studies.

With our study being conducted with various deep learning techniques, we also provide comparative analysis details of which techniques perform better in our study as it enhances the usability of different techniques. An adequate research has been done on these approaches RNN (A recurrent neural network is a type of artificial neural network commonly used in speech recognition and NLP) and hybrid NLP (basic transfer learning) to perform our classification and provide a comparative analysis which is the most significant and fruitful chapter in our study that has been shown in the chapter of result analysis.

**1.3 Contribution**

From all these studies we mentioned above we can decide that these studies have been promising research studies over the decade where authors approach DL or other basic ML approaches where study shows better results in contribution to our future field. In our study, we performed a multi-class text classification where we conduct our studies through basic ML and DL, also through feature extraction. The practice of immediately transforming data into numerical features that may be handled while retaining the original data set's information is known as feature extraction. It yields better outcomes than just applying machine learning to raw data. Word2vec is a two-layer neural network that vectorizes words in natural language processing to evaluate text. It takes a text corpus as input and returns a set of feature vectors representing the words in the corpus. Word2vec is not a deep neural network, but it does transform text to a numerical representation that deep neural networks can understand. Word2vec is a piece of software that learns word associations from a big corpus of text using a neural network model. After being trained, a model like this may distinguish synonyms and recommend new words for a sentence.

Our study has been done on the process where the dataset is to be classified in 4 different categories which are Computer science, Mathematics, Physics, Statistics. In order to perform such classification, we used abstract from papers as input and then these abstracts are used to perform feature extraction where using different NLP techniques and other machine learning processes to find unique words and similar words with help of corpus where labelize for every word.

We approached the transfer learning method to conduct our study method. Transfer Learning is a method of machine learning in which we use a pre-prepared model as a starting point for a new task. Simply put, the model trained in one task, considers the problem as an optimization that allows you to quickly improve while modeling the second task. Transfer training that uses the network [2]. However for a text category for which type data and transition training is of no use. Although many researchers have proven that transfer learning works well when the source dataset and target for transfer learning contain similar types of texts and labels.

In this study, we also performed a comparative study on feature extraction. Feature Extraction can be a challenge for textuals as we need to find through some designated parameters what the features would be.

As they are know feature for textual data, we set our parameters for the study we are conducting for sentiment analysis we targets keywords such as adjectives, affirmative or positive or negative but in our study case need features that set our data different from others so we need to some sort of unique words that can segment our class system for our papers, such as AI, DL would the keywords for deciding or segmenting the papers in Computer Science category, our study covers a comparative process three different feature extractions in order to perform such study and we used TF-IDF to vectorize out whole inputs for our baseline models.

The comparative studies are done on GloVe, Word2Vec, Word Sequence. We have used word sequence as a function of feature extraction for Deep Learning algorithms, where each word in the vocabulary is connected with an N-dimensional vector of real values. To accomplish this, we first used the 'tokenizer.text' module from the 'tensorflow.keras.preprocessing.text' package to tokenize the documents into words. Next, we created a word dictionary called 'word\_index' containing each unique word in the dataset, where word value is the index number of the dictionary and number of words parameter was set to 70,250, which is the total number of unique words in the dataset.The 'Word2Vec' module from 'gensim.models' was used to train the model in CBOW (Continuous Bag of Words Model) for the purpose of extracting feature vectors from textual input using the Word2Vec methodology. The train dataset was transformed into lists of tokens named 'sentences' and the model was trained with the word vector dimension set to 128. Following this, the feature vector was transformed into a [70255x128] matrix for use as the weights matrix in the RNN's embedding layer. GloVe, developed by Global Vectors, is a distributed model of word presentation. This model is an unaccompanied learning algorithm to obtain a vector representation of words. The distance between words is obtained by comparing words in a significant place according to semantic similarity.GloVe is an unsupervised learning method for obtaining vector representations of words. The global corpus is exercised literally with Corpus in statistics, and the obtained representations show an interesting linear infrastructure of vector space.

LSTM refers to long-term memory networks used for deep learning. He can study the long-term dependencies of different serial neural networks (RNNs), especially in sequencing problems. Two-way long-term memory (bi-lstm) is a process in which each neural network carries sequential information back (from future to past) or forward (from past to future) in both directions. In both directions, our input current flows in two directions, which distinguishes two lstm from a normal lstm. GRU is one of the most well-known gated RNNs that have been used to address the fundamental challenges of vanishing gradient and expanding gradient in basic RNNs; it also shows modeling's extraordinary ability to capture long-term relationships between sequence part.A duplex GRU or BiGRU is a sequential processing model consisting of two GRUs. One in the forward direction and the other in the backward direction. It is a continuous two-way neural network that has only a gateway of input and forgetting. With this comparaison in our study we are able determine which algorithm works better for Natural language processing techniques .

Chapter 2

# Literature Review

**2.1 Background Studies**

## Over the years, the classification problem has been a prominent topic among researchers. They have been employing various methodologies and techniques to handle classification problems for years, including text classification, sentiment analysis, image processing, and email filtering.

**2.1.1 Machine Learning**

Machine learning is the study of algorithms that learn from data to enhance statistical models to execute a specific task without explicit instructions.

Because each machine learning algorithm has its unique set of strengths and drawbacks, no single algorithm is suitable for all classification scenarios. On specific problems and datasets, a specific machine learning algorithm works well. We employed two separate supervised machine learning algorithms, Support Vector Machine and Random Forest, to perform the classification to conduct our study.

## **2.1.1.1 Support Vector Machine**

The support vector machine is a supervised classification method that uses a hyperplane to distinguish between classes with the highest marginal distance, where the margin is the distance between sample points and the hyperplane [4]. Sample points from the margins are defined as support vectors [6]. The purpose of employing a large marginal distance is to reduce generalization error since a larger marginal distance indicates a lower generalization error, whereas a small marginal distance indicates an overfitted model [1].

Assume that the decision hyperplane is parallel to the model's negative and positive class margins. This can be expressed as,

|  | (1) |
| --- | --- |
|  | (2) |

We could get, by subtracting the equations,

|  | (3) |
| --- | --- |

Normalizing the equation by the length of the vector w,

|  | (4) |
| --- | --- |

So we could arrive at the following equation,

|  | (5) |
| --- | --- |

The marginal distance, which we must maximize by maximizing and under the condition that samples are correctly classified, can be stated as,

|  | (6) |
| --- | --- |
|  | (7) |

Where, the number of samples in the dataset is denoted by N.

All positive samples should be positioned behind the positive hyperplane, while all negative samples should be positioned on the opposite side of the negative hyperplane. It is easier to decrease the reciprocal term in practice which is [7].

Because linear constraints must be relaxed for nonlinearly separable data in order for the optimization to converge in the presence of misclassification under acceptable cost penalization, a variable called slack is introduced to measure the level of constraint violation.[3] Then the equation can be written as, after adding positive slack value to the previous equations,

|  | (8) |
| --- | --- |
|  | (9) |

So objective function to be minimized becomes,

|  | (10) |
| --- | --- |

Where C is a defined regularization parameter that determines the tradeoff between maximizing the margin and reducing the training error. A high value of C correlates to a severe error penalty, whereas a low value of C implies leniency for misclassification error. Using the C parameter to modify the width of the margin and therefore adjust the bias-variance trade-off, reducing the value of C increases the bias and decreases the model's variance [3].

# **2.1.1.2 Random Forest**

Random forest is an ensemble classifier in which many decision tree classifiers are integrated using the bagging approach, which decreases the variance of a decision tree [7]. Here, each tree votes for the most popular class to classify a Vector input [8]. The fundamental concept is to design a model to avoid overfitting issues by averaging numerous deep decision trees and use their high variation in performance. There are several methods for selecting characteristics for decision tree induction, and the majority of these methods provide a quality measure directly to the attribute. The Information Gain Ratio criteria [9] and the Gini Index [10] are two examples. The random forest classifier employs the Gini Index, which assesses the impurity of an attribute relative to the classes, as an attribute selection measure [11].

For a given training set T, selecting one sample at random, assuming that the sample’s class , the Gini index can be written as [12] :

|  | (11) |
| --- | --- |

Where is the probability that the selected case belongs to class .

One of the key advantages of random forest classifiers is that fully developed trees are not pruned when they reach their maximum depth [12]. Choice of pruning techniques influences the performance of tree-based classifiers, but not attribute selection measures [13], and generalization error always lowers as the number of trees increases, even without pruning. The two essential predefined parameters are the amount of features employed at each node and N, the number of trees to be produced. Consequently, if the random forest classifier consists of N trees, each sample is delivered to each of the N trees, which then selects the class with the most votes out of N votes [8].

**2.1.2 Deep Learning**

In a variety of text classification tasks, such as sentiment analysis, news categorization, question answering, and natural language inference, deep learning–based models have outperformed classical machine learning–based approaches. For text categorization there are various types of classic techniques of deep learning models including feed forward neural network, RNN based model, CNN based model, graph neural network, Siamese neural networks and many more.

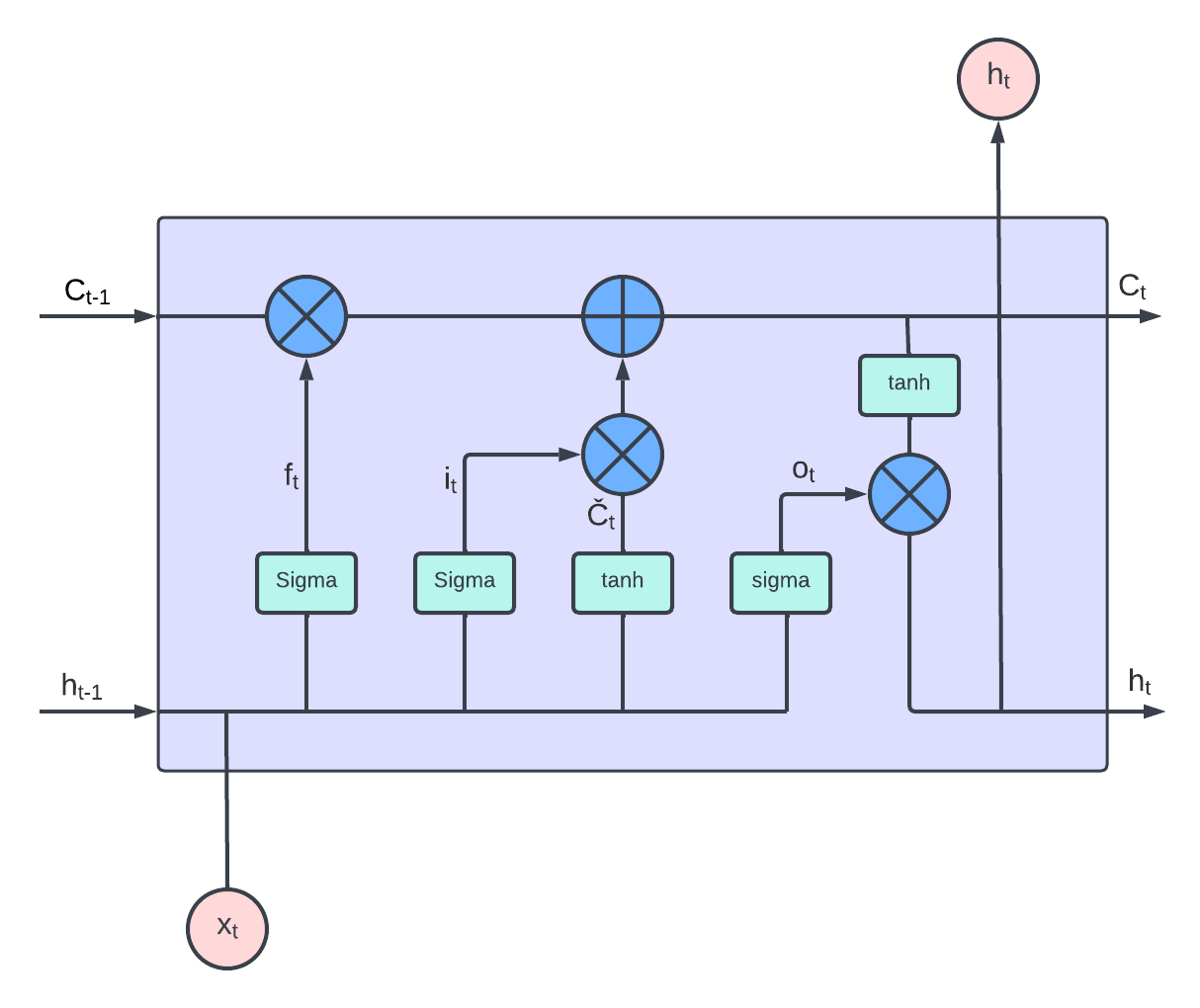
**2.1.2.1 Word Embedding**

Word embedding is a learnt representation for text in which words with the same meaning are represented in a comparable manner [17]. Individual words are represented as real-valued vectors in a predetermined vector space using word embeddings where words are stored in a high-dimensional space, with similarity in the vector space corresponding to semantic similarity. Each word is mapped to a single vector, and the vector values are learned in a neural network-like manner [16]. This is useful when using neural networks and deep learning models to solve natural language problems when numbers are required as input [3]. Features of the input texts are extracted from these embedded vectors using feature extraction techniques such as Bag of Words, TF-IDF, GloVe, Word2Vec, Word2Seq etc and feed the features to the embedding layer of the deep learning model.

**2.1.2.2 LSTM (Long Short Term Memory)**

RNN-based models treat text as a sequence of words and are designed to capture word dependencies and text structures for text classification, whereas LSTM is a popular architecture that is intended to better capture long-term dependencies [20]. LSTMs are specifically designed to circumvent the problem of long-term dependency. There are various varieties of LSTM [21] with modest alterations, but every LSTM design requires four components to regulate the flow of information into and out of the cell: cell state, forget gate layer, input gate layer, and output gate layer [22].

The cell state traverses the whole chain in a straight path, with only modest linear interactions with the gates. As LSTM cannot add or delete information in this cell state, gates consisting of a sigmoid neural network layer and a pointwise multiplication operation are inserted.

****

**Figure 2.1.2.2:** LSTM model

The initial phase of LSTM is for a sigmoid layer known as the "forget gate layer" to determine which cell state information must be forgotten. It takes , and outputs a number between 0 and 1 for each number in the cell state . LSTM follows equation (1-6) to train the model on word embedding The operation here is,

|  | (12) |
| --- | --- |

Where is the weight [] and is the bias.

In the subsequent phase, it determines which data will be saved in the cell state. In the first section, the input layer that determines which values should be updated is a sigmoid layer. A tanh layer is then applied, which generates a vector of new candidate values, , which may be added to the state. Finally, these two layers are pointwise multiplied to generate an update to the state.

|  | (13) |
| --- | --- |
|  | (14) |

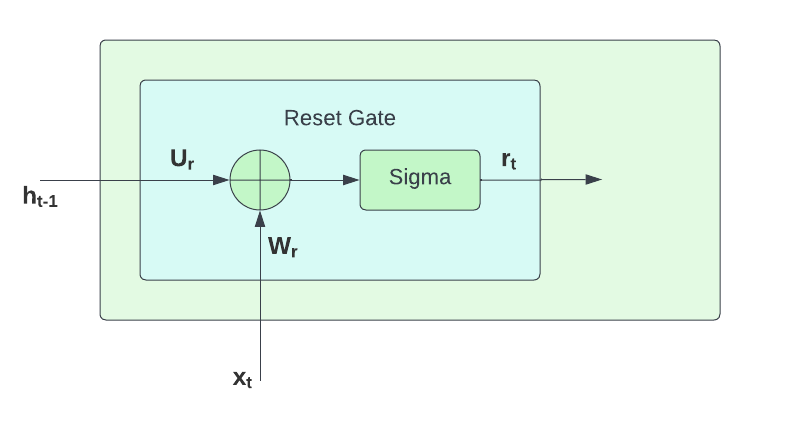
thereafter, updating the old cell state with the new cell state, it multiplies the old state with for forgetting and adds \* for adding information that has been decided to add.

|  | (15) |
| --- | --- |

The output must then undergo two operations: first, a sigmoid layer that determines which portion of the cell will be output, followed by a tanh layer whose output is multiplied with the sigmoid layer's output.

|  | (16) |
| --- | --- |
|  | (17) |

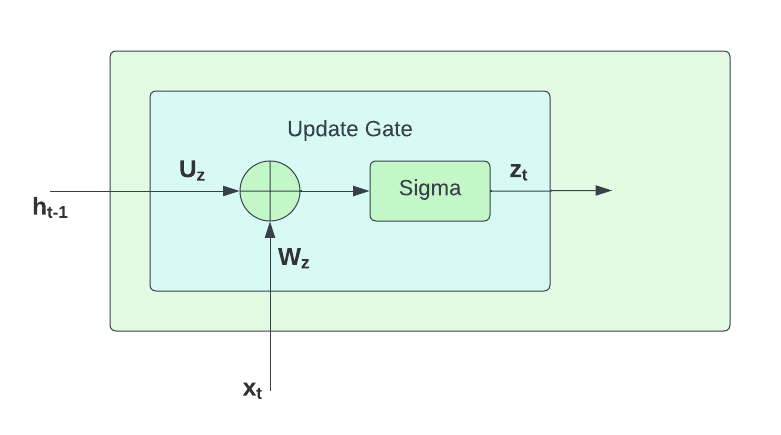
**2.1.2.3 GRU (Gated Recurrent Unit)**

GRU is one of the most prominent gated RNNs that have been employed to overcome the fundamental issues of vanishing gradient and exploding gradient problem in basic RNNs; it also exemplifies the remarkable capacity of modeling to capture long-term dependencies between the elements of a sequence [23]. It combines the input and forget gates of LSTM into a single "update gate" and includes a separate "reset gate." Unlike LSTM, GRU does not require a memory cell and has fewer parameters [28]. 

**Figure 2.1.2.3.1:** GRU Reset Gate

In this instance, the reset gate is responsible for merging the new input with the prior input and functions according to the following equation:

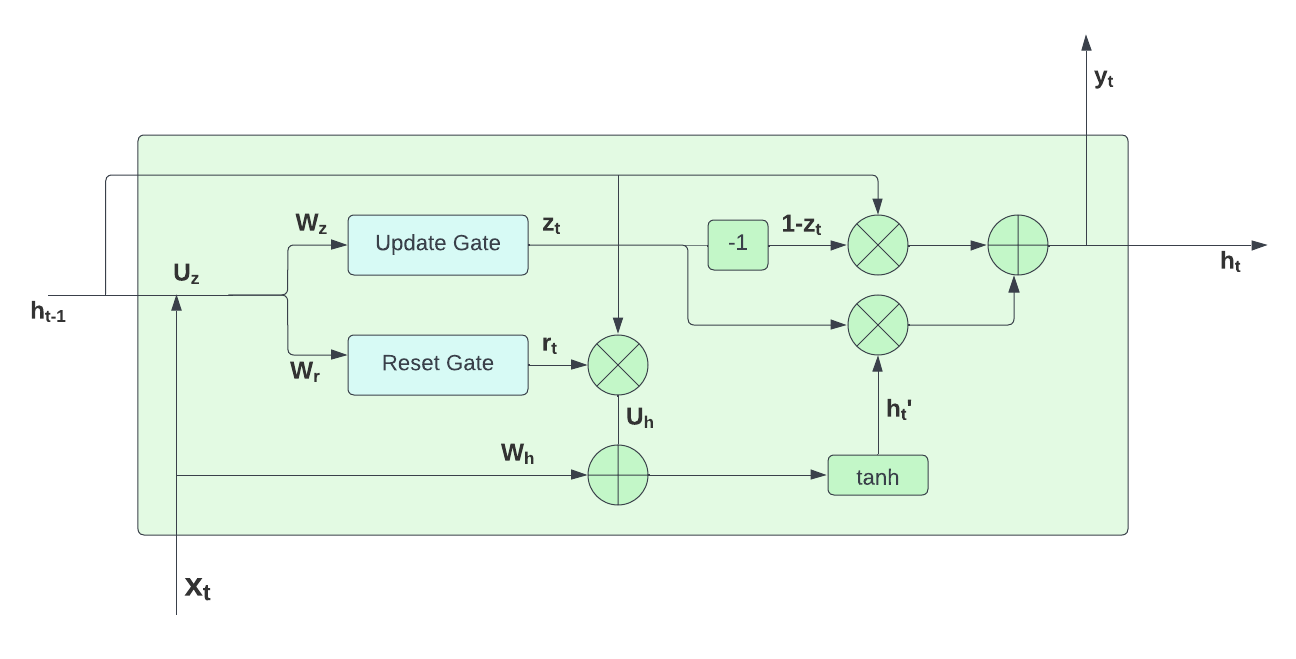
|  | (18) |
| --- | --- |

Where, is the reset gate vector, [ , ] are weights matrix, is a sigmoid activation function, is the input vector, is output vector of previous neuron and bias is . 

**Figure 2.1.2.3.2:** GRU Update Gate

The update gate determines how much of the prior memory must be saved [26], and the equation for the update gate vector is as follows:

|  | (19) |
| --- | --- |

At the state t, calculation of the new state is calculated by the following equations,

**Figure 2.1.2.3.3:** GRU

|  | (20) |
| --- | --- |
|  | (21) |

Where, is candidate activation vector, is the output vector, [, ] are weights matrix and tanh is the hyperbolic tangent. The operator ,‘’ refers to the hadamard product of two matrices.

**2.1.2.4 Bi-directional Models**

In both Bi-LSTM and Bi-GRU, sequences of input data are sent to the models in two distinct ways. The Bi-GRU contains the forward GRU, which reads the sentence from step 0 to t and the backward GRU, from step t to 0 [26].

|  | (22) |
| --- | --- |
|  | (23) |

The Bi-LSTM model also operates with the same technique as Bi-GRU, has the forward LSTM, and the backward LSTM, .

|  | (24) |
| --- | --- |
|  | (25) |

Then, combine the two outputs for the final result,

|  | (26) |
| --- | --- |

**2.1.4 Performance Evaluation Metrics**

**2.1.4.1 Confusion matrix**

In machine learning, the confusion matrix is the most compelling instrument for predictive analysis. The confusion matrix is often used to assess the effectiveness of a classification-based machine learning (ML) model. This matrix is composed of columns and rows that list the number of instances as absolute or relative "actual class" vs "predicted class" ratios.

**Table 2.1** Example of confusion matrix

| **Total Population** | **Predicted Positive** | **Predicted Negative** |
| --- | --- | --- |
| **Positive** | *True Positive (TP)* | *False Negative (FN)* |
| **Negative** | *False Positive (FP)* | *True Negative (TN)* |

A True Positive (TP) is an outcome for which the model successfully predicted the positive class, whereas a True Negative (TN) is an outcome for which the model correctly predicted the negative class. False Negative (FN) indicates that the case was positive but the prediction was negative, whereas False Positive (FP) indicates that the case was negative but the prediction was positive.

**2.1.4.2 Precision**

Precision explains how many positive classes the classifier has predicted correctly from all the predicted positive classes. It determines the model's reliability, with False Positives being of more concern than False Negatives.

|  | (27) |
| --- | --- |

**2.1.4.3 Recall**

Recall assesses the number of accurately predicted positive classes based on the actual positive classes. It is beneficial in situations when False Negative poses a higher concern than False Positive.

|  | (28) |
| --- | --- |

**2.1.4.4 F-Score**

The F-score is the harmonic mean of precision and recall. When two models have poor precision and high recall, or vice versa, it is difficult to compare them; consequently, F-score may be used to overcome this problem.

|  | (29) |
| --- | --- |

**2.1.4.5 Accuracy**

Accuracy is the proportion of true results (both true positives and true negatives) among the total number of cases examined. It describes the frequency of which the model predicts accurate outcomes.

|  | (30) |
| --- | --- |

**2.2 Related Works**

Research paper classification is basically a different level of text categorization and tremendously important in the field of machine learning and deep learning. Natural language processing is the core idea working behind it. The whole task has been handled differently in the previous research in order to make it smoother and time saving so that it paves the way to improve the classification in near future.

Bharath Kandimalla et al. (2021) illustrated about Deep attentive neural networks (DANN) in their work which classifies papers extracted from scholarly works based on their abstracts. The dataset is trained using abstracts from web of science (WoS), containing almost 45 million elements. They have been categorized in three labels: science, social science, Art and literature. As baselines, Random Forest (RF), Naïve Bayes (NB, Gaussian), Support Vector Machine (SVM, linear and Radial Basis Function kernels), and Logistic Regression (LR) are used and 1014 dimensional vector has been used to model the abstracts. The result they got using DANN is worth mentionable and almost 10% better than the best machine learning model LR. In terms of Neural network methods, they trained FastText or GloVe and piled them with BiGRU or BiLSTM and attention mechanisms to get the best result onwards. Classifier performance has been improved by retraining WE models. It has been concluded stating that if neurons and layers are added then it may have little effect on the performance of the classifier [42].

Tan Yue et al. (2022) worked on PaperNet-Dataset having 12 datasets including multi-modal data, containing 2 coarse-grained (CV and NLP) and 20 fine-grained (7 in CV and 13 in NLP) classes. Samples were collected from Google Scholar. Almost 38000 samples are used from renowned NLP and computer vision conferences. The proposed method is a combination of the complementary strengths of MobileNetV3 and Albert for better multi-modal information joint representation. A comparison of the proposed model and some previously developed good embedding and classification method has been shown through the algorithm presentation part such as ResNet50, DenseNet121, MobileNetV3, ULMFiT, Albert, Concat. The proposed method gained the highest accuracy in PaperNet\_2, PaperNet\_20, PaperNet\_NLP so according to the authors this can be used as a benchmark in near future [43].

Monir ECH-CHOUYYEKH et al. (2019) proposed the method of using convolutional neural network for classification task of articles and showed a definite comparison of results of CNN with other ML algorithms like as the Bayesian network, linear regression, Support vector machines (SVM). Word embeddings are used to extract text characteristics linking up each word or sentence with a N dimensional vector of real numbers. Word2vec, GloVe and FastText have been shown in this method. However, they worked on the dataset of “web of Science Dataset '' (2017) containing 35,238 scientific articles. Keras library implementation is shown in this method. The result part illustrates that CNN algorithm combined with the mentioned model worked really well, obtaining the classification rate of 82% which is definitely one of the best results [44].

Farman Ali et al. (2019) have shown a word embedding approach on sentiment analysis which is based on ontology and Latent Dirichlet Allocation (LDA). Through this system transportation contents are gathered using social platforms and LDA is used in order to use them for analysis purposes. This method is tremendously successful as it receives accuracy of 93%. Web ontology and JAVA are working behind this model. They have noted that topic2vec along with the proposed ontology extract features cutting inappropriate words off and hence reploting a meaningful document. Word2vec which is another popular process has also been used in their word embedding process. The performance of string2vec, word2vec, doc2vec, glove2vec, and lexicon2vec are compared with the proposed model to evaluate their task. Classification of data has been conducted using machine learning algorithms and further results have been listed through various comparisons. In terms of deep learning, golve2vec played an important role here. Data has been collected from social media like facebook, twitter and also from news articles, using TripAdvisor. In working with these, they used news articles published between April 2017 and July 2017. In the case of TripAdvisor, 1851 reviews about New York and London have been included and have an average of 73 words per review. Facebook data were extracted Graph API, Java client, RestFB using various pages and hence posts published between March 2016 and January 2018 have been used. From twitter, around 30,000 tweets regarding transportation are fetched. The accuracy was reliable in comparison with string2word and glove2vec and moreover the proposed method has a very low complexity. It is noted in their work that they have used Waikato Environment for Knowledge Analysis (Weka) to conduct the evaluation process hereby using different classifiers [45].

Suneera C M et al. (2020) showed different machine learning and deep learning algorithms to classify texts. As a dataset, they used 20 newsgroup datasets, each article containing almost 400 characters. Basically it contains 18846 documents based on subjects like politics, religion, computers, sports etc. TF-IDF, Word2Vec, and the state of the art language model BERT (Bidirectional Encoder Representations from Transformers). TF-IDF worked best in their findings rather than word2vec and BERT embeddings to classify articles. Logistic Regression gained 82.74% accuracy in classical ML algorithms. LR proves itself as the best ML algorithm and Bichannel Convolution Neural Network model achieved the highest place in deep learning method [46].

Yuanchao Liu et al. (2013) initiated work based on a support vector machine which is a semi-supervised method referred as Transductive Support Vector Machine. The main task is to classify scientific papers based on their abstracts, dividing them into four categories such as: the background, the goal, the method and the result. As the existing data was not enough, they made the compilation and marked the instances for conducting the work. In this way, they were able to make a good outcome in terms of accuracy which is around 75.86%. The process has been conducted in 3 steps which includes building the corpus of abstracts(analyzing, tagging), applying various experiments on these so that a better feature vector can be gained and finally carrying out the task using supervised and semi supervised methods. The papers that are used as dataset have been owned through the website (http://www.sciencedirect.com/).

A scale of 4550 abstract sentences having four labels from the corpus has been used resulting in 127718 words after splitting sentences. The corpus was divided into two parts: the training set contained 2404 labeled sentences and the testing set contained 2146 labeled sentences. The SVM is trained using libsvm-3.16. They applied a CHI test for the categories. Joachim's (1999) proposed a semi-supervised learning method named TSVM which deals with labeled and unlabeled data. The accuracy increased with unlabeled data [47].

Shovan Chowdhury et al. (2020) showed various kinds of machine learning algorithms in order to classify contents. Text classification is conducted which is basically research paper classification on subjects like science, business and social science. In this work again abstract has been used like many other models to continue classification as abstracts reflect the whole work with a few words. TF-IDF and Bag of Words are used for vectorization. For classification, Support Vector Machines (SVM), Naïve Bayes, K-Nearest Neighbor (KNN), and Decision Tree classification algorithms are included [48].

Sang-Woon Kim and Joon-Min Gil (2019) have done the same research work. 107 research abstracts are compiled to make the dataset where science and social science class consists of 36 abstracts and business class consists of 35 abstracts. Online sources such as Google Scholar, Research Gate etc. are used for this purpose. Training session consists of two thirds of the data and the others are for testing purposes. In terms of result accuracy SVM worked the best whereas Decision tree worked worst. KNN and Naïve Bayes worked a little better and stayed in between. If more data is added, the algorithm may work better according to them [49].

Tran Thanh Dien et al. (2019) represented a model that can categorize articles by extracting information. KNN, support vector machine, Naïve Bayes these are used to pre-process data and classify them initially. They found out the accuracy was almost 91% and concluded declaring SVM as the best for classification. In this system, after they give an input the model will extract author, title and abstract in order to conduct the task of classification. Being a preformatted document, it will be easier in this system. For word segmentation, the VnTokenizer segmentation tool has been used in this method. The Experimental datasets were 680 scientific articles (10 topics) and 10,000 articles of newsletters, published in Can Tho University Journal of Science from 2016 to 2018. Another important task here they have done is converting docx. To txt. Files before pre-processing word segmentation. Using SVM algorithm the accuracy that has been gained was more than 90% and through naïve bayes it was almost 87.6% [50].

I Jaya, I Aulia et al. (2019) represents SVM as a useful method in the field of document classification like many others conducting research in this field. They took 150 documents as data samples and 50 documents as data testing materials. Based on metadata which inputs the title, researcher’s name and keywords and again abstracts have been used to classify documents. After preprocessing data, TF-IDF has been used here like many other methods. For the dataset, data is taken from International Conference on Computing and Applied Informatics (ICCAI) and Springerlink collection and only the abstract part is being used. In pre-processing, tokenization, removing stop words and stemming is done. Abstracts from five categories such as: : Computer Graphics and Image Vision, Computer Systems, Data Science, Human Computer Interaction and Information Security achieved the accuracy of 90% after classification through this method [51].

Chapter 3

# Methodology

**Figure 3.1:** Work-flow

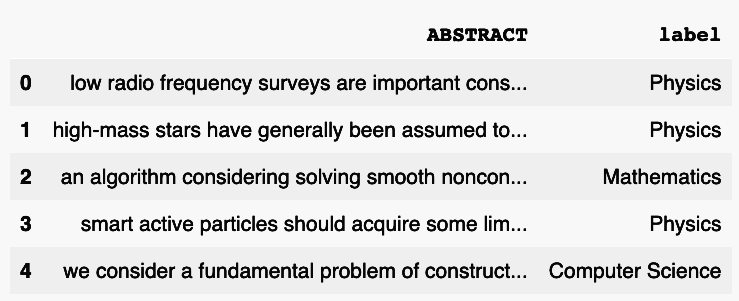
**3.1 Data collection**

The goal of our study is to categorize research article abstracts into relevant categories. Figure 3.1 depicts the entire work procedure. Our initial goal is to collect abstracts from research papers in four distinct fields: computer science, mathematics, statistics, and physics. We gathered a dataset from the Hackathon that included the abstract parts of 20,006 research papers from a variety of subject areas, organized in a multi-class and multi-label format and made publically available for future research.

**3.2 Data Description**

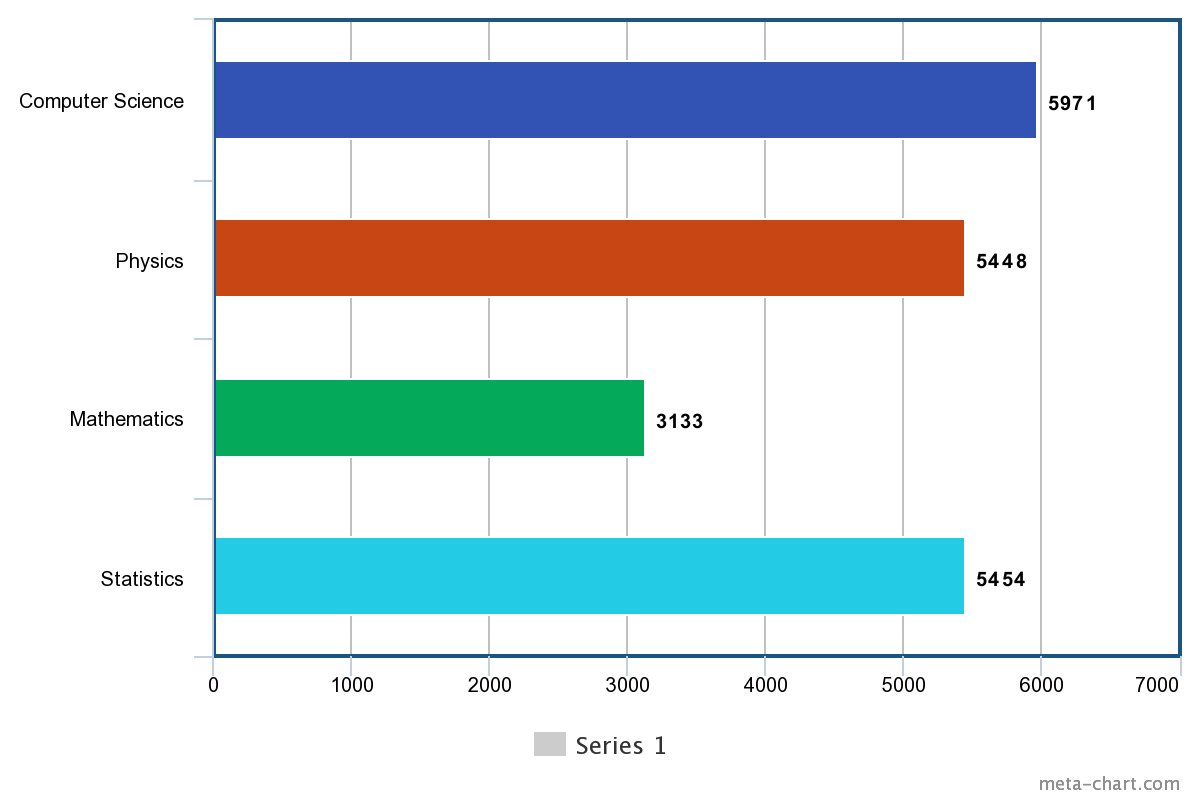
Each sample is labeled as either 'Statistics,' 'Physics,' 'Mathematics,' or 'Computer Science,' as well as 25 subdomains of these categories in a multi-label fashion. The subdomains are ‘Analysis of PDEs’, ‘Applications’,’Artificial Intelligence’,’Astrophysics of Galaxies’,’Computational and Language’,’Computer Vision and Pattern Recognition’, ‘Cosmology and Non Galactic Astrophysics’, ’Data Structure and Algorithm’, ‘Differential Geometry’, ‘Earth and Planetary Astrophysics’, ‘Fluid Dynamics’, ‘Information Theory’, 'Instrumentation and Methods for Astrophysics', 'Machine Learning', 'Materials Science', 'Methodology', 'Number Theory', 'Optimization and Control', 'Representation Theory', 'Robotics', 'Social and Information Networks', 'Statistics Theory', 'Strongly Correlated Electrons', 'Superconductivity', 'Systems and Control'.

We removed the subdomain columns and the 'id' column from the dataset in order to do multi-class classification since we only performed supervised multi-class classification and compared the results of the different algorithms.



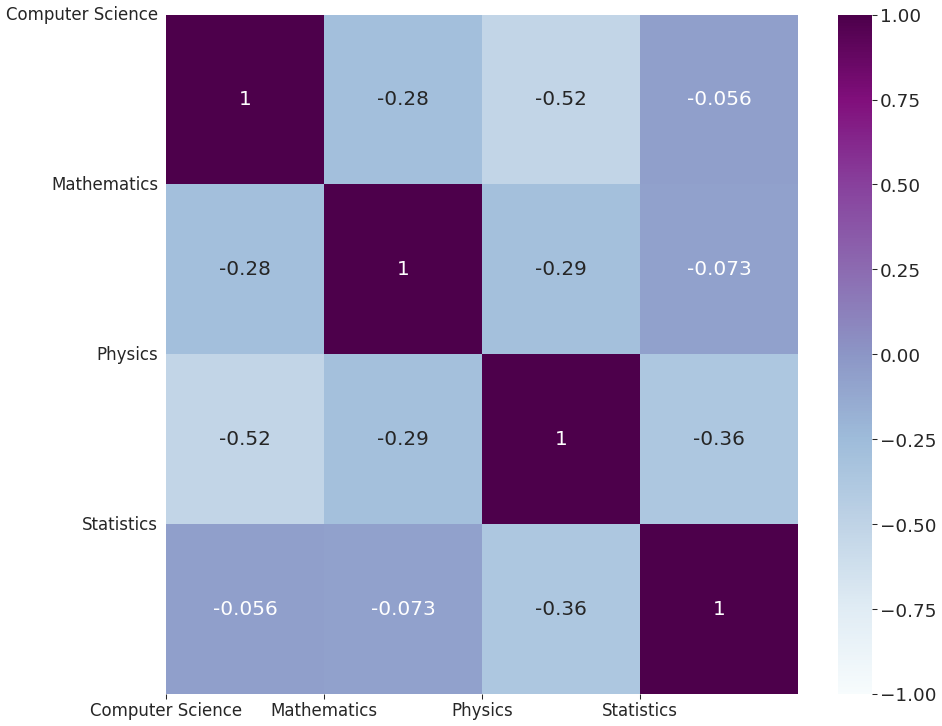
**Figure 3.2.1:** Label Encoded Dataset

**3.3 Data Analysis:**

The overall percentages and numbers of the classes are shown in Figure 3.3.1 The 'Computer Science' class makes up 29.8% of the dataset sample, whereas 'Mathematics' consists 15.7 percent, 'Statistics' makes up 27.3 percent, and 'Physics' covers up 27.2 percent.

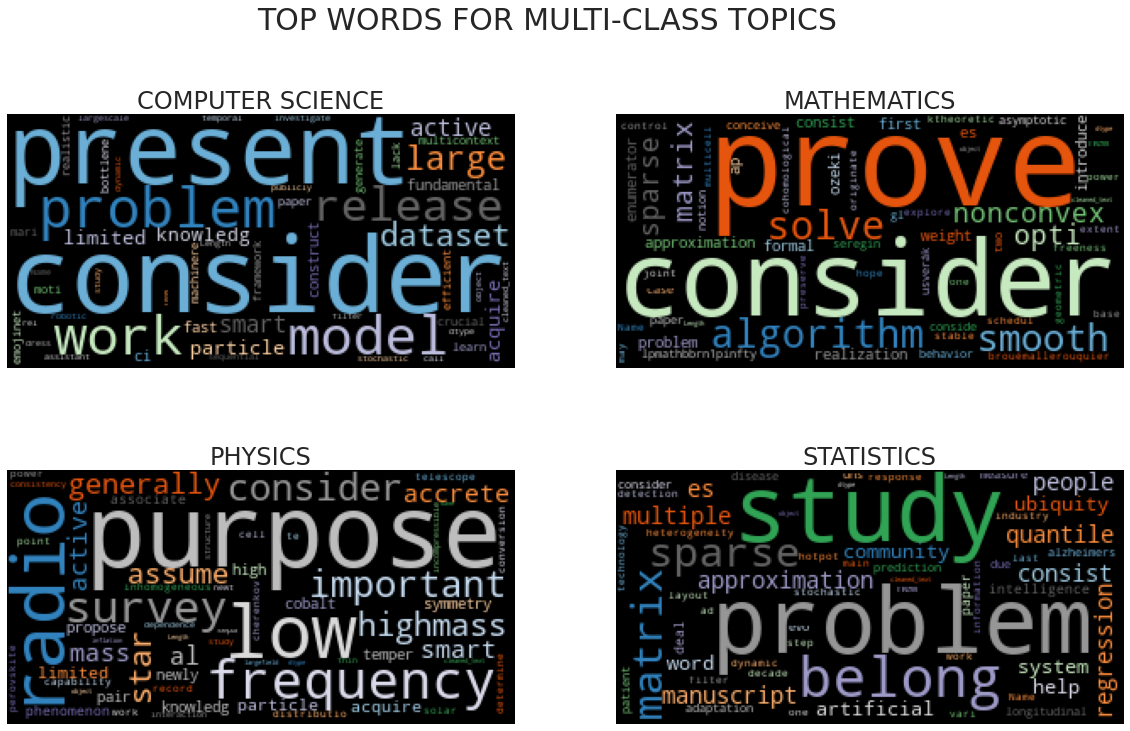
**Figure 3.3.1:** Data Distribution

The correlation matrix between the multiclass labels, shown in Figure 3.3.2, shows the correlation coefficients for various variables over all possible pairs.



**Figure 3.3.2:** Correlation Matrix

Figure 3.3.3 depicts the dataset's word clouds based on the categories.

​​

**Figure 3.3.3:** WordCloud

**3.4 Data Pre-Processing :**

Before beginning the classification procedure, preprocessing is necessary. Because the dataset we are dealing with may contain noise such as misspelled words, poor punctuation, non-standard abbreviations, erroneous capitalization, and grammatical errors. It may contain punctuation, stop words, special characters, and numerical values, all of which are irrelevant to our suggested classification task. Therefore, the dataset must be processed before classification in order to improve the outcome of the task.

**3.4.1 Text Transformation:**

**Contractions expanding:** Words or combinations of words that are abbreviated by eliminating letters and replacing them with an apostrophe are known as contractions. We expanded the dataset's existing english contractions as a first stage in the preprocessing procedure. We used the internet to find english contractions and their expansions, which we then applied to our dataset. "Ain't" will become "are not" after running the function, whereas "we'll" will become "we will."

**URL and Punctuation Remove:** URLs may include information extraneous to our categorization procedure. Therefore, we eliminated the URLs because they did not give any useful information. In addition, because the models are unable to understand punctuation, the presence of punctuation makes the text noisier. Regular expressions were employed to identify punctuation marks and URLs, which were then replaced with whitespace and eliminated.

**New line and Whitespace Remove:** Some abstract parts of research articles may comprise more than one paragraph, resulting in an unnecessary newline. A string may have additional whitespace at the beginning and end of a phrase, and it may also contain several whitespace instead of a single one. We utilized regular expressions to locate and remove the new line. We utilized the.strip() function to eliminate start and end whitespace, and we solved the multiple space problem by replacing several spaces with a single space.

**Converting the dataset into lowercase:** Because our algorithm recognized 'The' and 'the' as two separate words, which is problematic for our purpose, we subsequently converted the dataset to lowercase characters. We converted all the capital characters to lowercase using the '.lower()' technique to solve this problem.

**3.4.2 Abbreviations removal:**

**Remove Stop-words:** Stop-words are a list of the most frequently used terms in a language, such as "the," "a," "but," "this," etc. Stop-words are frequently found in significant amounts in all samples, however they contribute little to no information. Therefore, we must reduce the quantity of low-level information in our text in order to reduce the system's wasteful calculation so that it can run more efficiently. As our dataset solely contains English text, we utilized the stopwords.words('english') function from nltk.corpus to remove English stopwords.

**Rare words removal:** There are unique terms in the collection, such as names, brand names, and product names, that appear relatively infrequently. We eliminated the top ten uncommon words by computing the frequency of each term in the sample.

**Lemmatizing and Pos-tagging:** For grammatical reasons, there will be many variants of a word in official documents. In our research, we employed lemmatization to reduce the inflection forms and derivationally related forms of a word to its root form. Stemming, which is routinely employed to finish the work, is another method. Stemmer operates on a single word without context, hence it is unable to distinguish between words with various meanings based on their parts of speech. In contrast, lemmatization considers the context and transforms words to their semantic roots, which is more accurate than stemming. The words were pos-tagged as Noun, Verb, Adjective, and Adverb before being lemmatized with WordNetLemmatizer ().

**3.5 Dataset Split:**

For our baseline techniques such as SVM and Random Forest, the dataset was divided into two datasets: training dataset (80 percent) and test dataset (20 percent). For deep learning algorithms, the training dataset was further subdivided into train dataset (88 percent) and validation dataset (12 percent ).

***Baseline approach:***

**Table 3.5.1:** Dataset Split of Baseline models

| **Dataset Name** | **Count** |
| --- | --- |
| Full dataset | 20006 |
| Training dataset | 16004 |
| Test dataset | 4002 |

***Deep learning approach:***

**Table 3.5.1:** Dataset Split of deep learning models

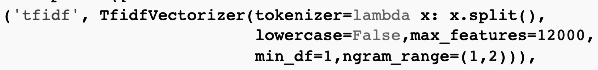
| **Dataset Name** | **Count** |
| --- | --- |
| Full dataset | 20006 |
| Train dataset | 14083 |
| Test dataset | 4002 |
| Validation dataset | 1921 |

**3.6 Feature extractions techniques:**

To study how different word embeddings influence categorization outcomes, we employ a number of popular methods. In this work, four distinct feature extraction algorithms, TF-IDF, Word Sequence, Word2Vec, and GloVe, are utilized to extract features from preprocessed text data.

**3.6.1 TF-IDF:**

For text data to be used as input in a Machine Learning algorithm, it must be in numerical form. The TF-IDF vectorizer is one of the most used methods for transforming text input into numerical form. The 'Tfdfvectorizer' package from ‘sklearn' was utilized to run TF-IDF on our dataset and extract the features. Where parameters including max feature = 12000, ngram range = (1,2), and min df = 1 are specified. The TF-IDF value has been utilized as input in machine learning algorithms SVM and Random forest.

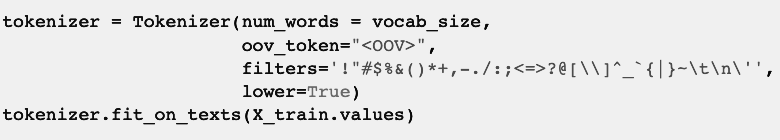


**Figure 3.6.1:** TF-IDF Vectorization

* max\_feature: Construct a vocabulary using just the top max\_features ranked by term frequency across the corpus.
* ngram\_range: The minimum and maximum range of n-values for distinct n-grams to be extracted.
* min\_df: When constructing the vocabulary, exclude words whose document frequency is strictly below the specified threshold.

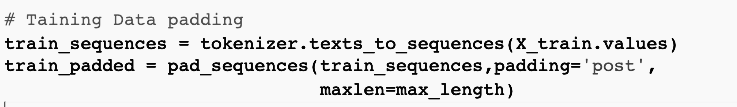
**3.6.2 Word Sequence:**

We have used word sequence as a function of feature extraction for Deep Learning algorithms, where each word in the vocabulary is connected with an N-dimensional vector of real values. To accomplish this, we first used the 'tokenizer.text' module from the 'tensorflow.keras.preprocessing.text' package to tokenize the documents into words. Next, we created a word dictionary called 'word\_index' containing each unique word in the dataset, where word value is the index number of the dictionary and number of words parameter was set to 70,250, which is the total number of unique words in the dataset. Then, we utilized the 'texts to sequences' module from 'tensorflow.keras.preprocessing.text' to represent the document as a one-dimensional vector containing the value of each word. Finally, the 'pad sequences' module from 'tensorflow.keras.preprocessing.sequence' was used to convert all sequence vectors to the same length, using the post-padding approach and setting the vector length to 200. The same procedure was applied to the training, validation, and testing datasets.



**Figure 3.6.2.1:** Tokenization

* num\_word: the maximum number of retained words depending on frequency. Only the num words-1 most frequent words will be retained.
* oov\_token: If provided, it will be added to the word index and used to substitute words that are not in the vocabulary during text to sequence calls.
* filters: Each element of this string represents a character that will be filtered from the messages. The default is all punctuation, tabs, and line breaks, excluding the ' symbol.
* lower: Whether to convert the texts to lowercase.



**Figure 3.6.2.2:** Sequence Padding

* padding: "pre" or "post" pad before or after each sequence, respectively.
* max\_len: length maximum of all sequences. Sequences will be padded to the length of the longest individual sequence if padding is not specified.

**Example:**

***Pre-processed texts:***

*“algorithm consider solve smooth nonconvex optimization problem propose inside worstcase take mathcaloepsilon32 iteration drive norm gradient objective function prescribe positive real number epsilon take mathcaloepsilon3 iteration drive leftmost eigenvalue hessian objective epsilon propose algorithm general framework cover wide range technique include quadratically cubically regularize newton method adaptive regularisation help cubics arc method recently propose trustregion algorithm contraction expansion trace generality method achieve introduction generic condition trial step require satisfy inside particular allow consider inexact regularize newton step use condition center around new subproblem approximately solve obtain trial step satisfy condition new instance framework distinct arc trace describe may view hybrid quadratically cubically regularize newton method numerical result demonstrate hybrid algorithm outperforms cublicly regularize newton method”*

***After applying texts\_to\_sequences:***

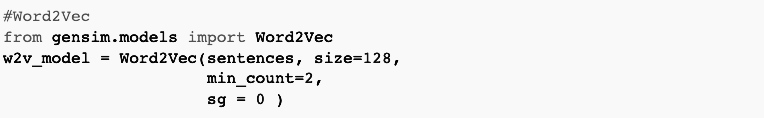
*[464, 238, 120, 7, 147, 917, 82, 11, 340, 688, 312, 502, 2, 3211, 4, 1068, 4, 29640, 506, 76, 269, 7, 824, 267, 3, 4, 3211, 2, 156, 9, 29641, 29642, 1148, 380, 3, 917, 291, 327, 26, 123, 9, 4, 29643, 506, 76, 63, 875, 319, 1421, 29644, 110, 29645, 28, 40, 922, 4, 3, 3211, 4607, 1924, 3, 135, 179, 447, 11831, 354, 618, 472, 981, 10, 3523, 2, 4505, 539, 938, 3251, 778, 2, 3910, 305, 447, 618, 3910, 931, 76, 305, 1065, 795, 365, 11, 18, 836, 1791, 1461, 726, 2496, 224, 3251]*

***After applying pad\_sequences:***

*[ 464 238 120 7 147 917 82 11 340 688 312 502 2 3211 4 1068 4 29640 506 76 269 7 824 267 3 4 3211 2 156 9 29641 29642 1148 380 3 917 291 327 26 123 9 4 29643 506 76 63 875 319 1421 29644 110 29645 28 40 922 4 3 3211 4607 1924 3 135 179 447 11831 354 618 472 981 10 3523 2 4505 539 938 3251 778 2 3910 305 447 618 3910 931 76 305 1065 795 365 11 18 836 1791 1461 726 2496 224 3251 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 0 ]*

**3.6.3 Word2Vec:**

The 'Word2Vec' module from 'gensim.models' was used to train the model in CBOW (Continuous Bag of Words Model) for the purpose of extracting feature vectors from textual input using the Word2Vec methodology. The train dataset was transformed into lists of tokens named 'sentences' and the model was trained with the word vector dimension set to 128. Following this, the feature vector was transformed into a [70255x128] matrix for use as the weights matrix in the RNN's embedding layer.



**Figure 3.6.3:** Word2Vec Model

* sentences: The iterable sentences are a collection of lists of tokens.
* size: Dimensionality of the word vectors.
* min\_counts: Ignores all words with total frequency lower than this.
* sg: training method, 1 for skip-gram, 0 for CBOW.

**3.6.4 GloVe (Global Vectors for Word Representation):**

In contrast to word2vec, which relies primarily on local information from words with local context windows, the GloVe method integrates word co-occurrence information or global statistics to determine the semantic associations between words in the corpus. This approach investigates the exact vector representation of words in comparable circumstances and constructs a co-occurrence matrix from corpus data by minimizing the average log-likelihood function [34]. GloVe learns the association between words by calculating the frequency with which terms appear in the same corpus [35]. The likelihood ratio of a word's occurrence has the potential to encode meaning and enhance performance on the problem of word analogies. In this work, a pre-trained word vector was obtained from [GloVe: Global Vectors for Word Representation](https://nlp.stanford.edu/projects/glove/) and utilized for the GloVe approach. The pre-trained model is trained in 300 dimensions using data from Wikipedia 2014 comprising 6 billion tokens and a vocabulary size of 400k. For the embedding layer of deep learning algorithms, it has been transformed to a [70255x300] matrix.

**3.7 Baseline models:**

**3.7.1 Support Vector Machine:**

SVM is a supervised classification technique that is effective for text classification with a large input dimension using text as features [38]. In this work, we employed the 'SVC' module from the 'sklearn.svm' library to do a non-linear classification using the tf-idf model as the classifier model's feature vector.

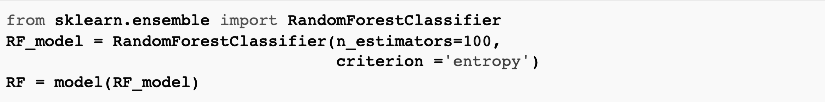


**Figure 3.7.1:** Support Vector Classifier

* random\_state: Controls the generation of pseudo-random numbers for shuffling data for probability estimations.
* kernel: Specifies the kind of kernel to be utilized by the algorithm.

**3.7.2 Random Forest:**

Using the 'RandomForestClassifier' module from the scikit learn' library with 100 n\_estimators and 'entropy' as the criteria function, we performed a random forest classifier. The tf-idf model's feature vectors were utilized to train the random forest model.



**Figure 3.7.2:** Random Forest Classifier

* n\_estimator: The number of decision tree classifiers in the forest.
* criterion: Split quality measuring function.

**3.8 Deep Learning models:**

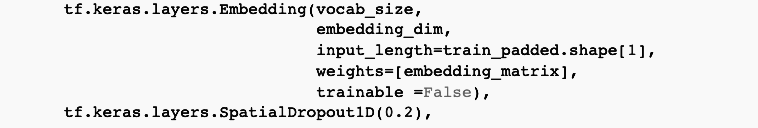
The padded version of the train dataset was used as the input for both bi-directional and directional deep learning models, where the RNN model was constructed using Keras's TensorFlow modules.

For constructing the embedding layer we set the parameters:

***GloVe embedding:*** input\_dim = 70256, output\_dim = 300, weights = embedding matrix extracted from GloVe feature extraction.

***Word2Vec embedding:*** input\_dim = 70250, output\_dim = 128, weights = embedding matrix extracted from Word2Vec feature extraction.

***Word Sequence embedding:*** input\_dim = 70250, output\_dim = 128, weights = padded sequence of train dataset.



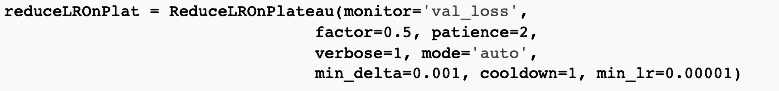
**Figure 3.8.1:** Embedding layer

* input\_dim: size of the vocabulary.
* output\_dim: dimension of the dense embedding.
* input\_length: Length of input sequences

For initializing LSTM and, GRU, the ‘tf.keras.layers.LSTM’ and ‘tf.keras.layers.GRU’ modules have been used where arguments such as ‘units’ was set to 128 and’ return\_sequences’ was set to True. The 'tf.keras.layers.Bidirectional' is used for Bidirectional LSTM and Bidirectional GRU. On both the LSTM and GRU layers, a 'Bidirectional' module has been implemented. Then, four dense layers were generated using the Relu activation function, a dropout of 0.5, and output space dimensions of 28, 128, 32, and 16 accordingly. The inputs were flattened using the 'tf.keras.layers.Flatten()' function. The last layer was created using a sigmoid function with the output dimension equal to the number of categories in our study, which was four.

The 'adam' optimizer was used to compile the RNNs, 'binary cross entropy' was used as the loss function, and 'binary accuracy' was used to measure the performance of the model. In each model, batch size was set to 128, epochs were set to 100, and validation data was set to a padded validation dataset.

To lower learning rate when a model stops progressing, the'reduceLROnPlat' module from the keras library was utilized, and a self-stopping model for accuracy was constructed with the aid of the keras library to terminate a model's training when the binary accuracy reached 99.90 percent.



**Figure 3.8.1:** Learning rate reducing function

* monitor: quantity to be monitored
* factor: by which the learning rate will be reduced.
* patience: number of epochs with no improvement after which learning rate will be reduced.
* min\_delta: threshold for measuring the new optimum, to only focus on significant changes.
* min\_lr: lower bound on the learning rate.

Chapter 4

# Result and Discussion

This section defines the experimental results that have been gained after applying the machine learning and deep learning algorithms. We also combined the analysis section in this so that a prominent comparison among these algorithms can be portrayed.

For analyzing the result of the proposed approaches, we evaluated the performance of the methods through traditional evaluation metrics such as confusion matrix, accuracy , precision, recall, F1 score and we have used the “Scikit learn” library to generate the results.

**4.1 Baseline models**

**4.1.1 Support Vector Classifier & Random Forest Classifier**

**Table 4.1** Performance of baseline models

| **Algorithms** | **Accuracy** | **Recall** | **Precision** | **F1** |
| --- | --- | --- | --- | --- |
| **SVC** | 0.823 | 0.826 | 0.824 | 0.825 |
| **Random Forest** | 0.844 | 0.844 | 0.841 | 0.843 |

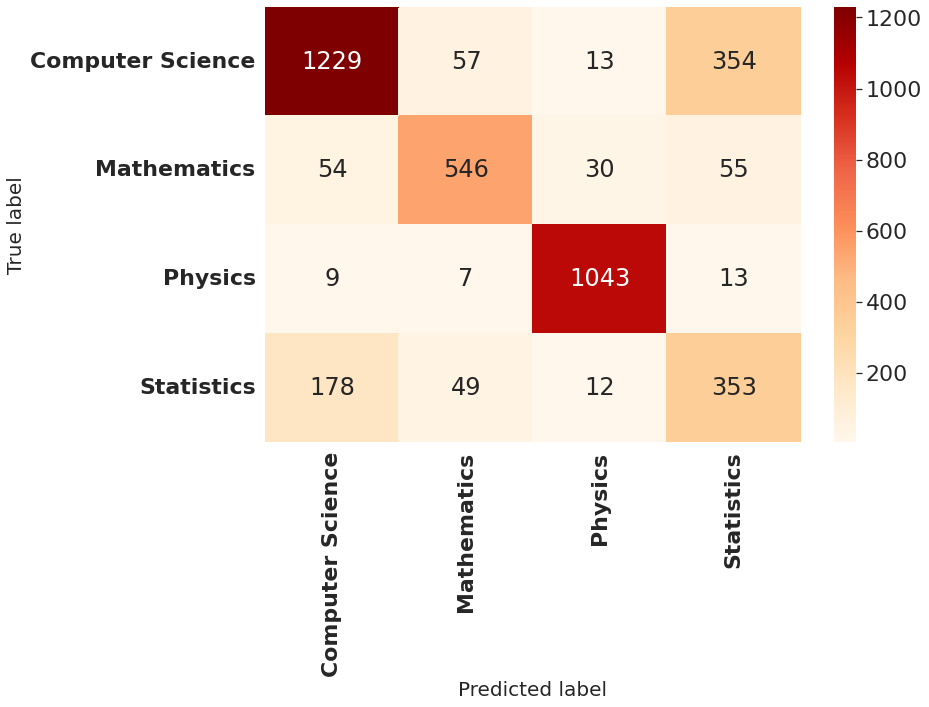
These are baseline algorithms of machine learning which were applied to the feature extraction process. We trained these models on TF-IDF, which extracted features from textual context but with presumably the result of accuracy, we can say, baseline algorithms do not perform better in any substance. As the table indicates, the performance of random forest model has significantly achieved better accuracy scores than support vector model. Random forest is an ensemble classifier in which many decision tree classifiers are integrated using the bagging approach, which decreases the variance of a decision tree [7]. Here, each tree votes for the most popular class to classify a vector input.

**4.2 Deep Learning**

**4.2.1 WordSeq Approaches**

**4.2.1.1 WordSeq + LSTM**

Confusion matrix:

****

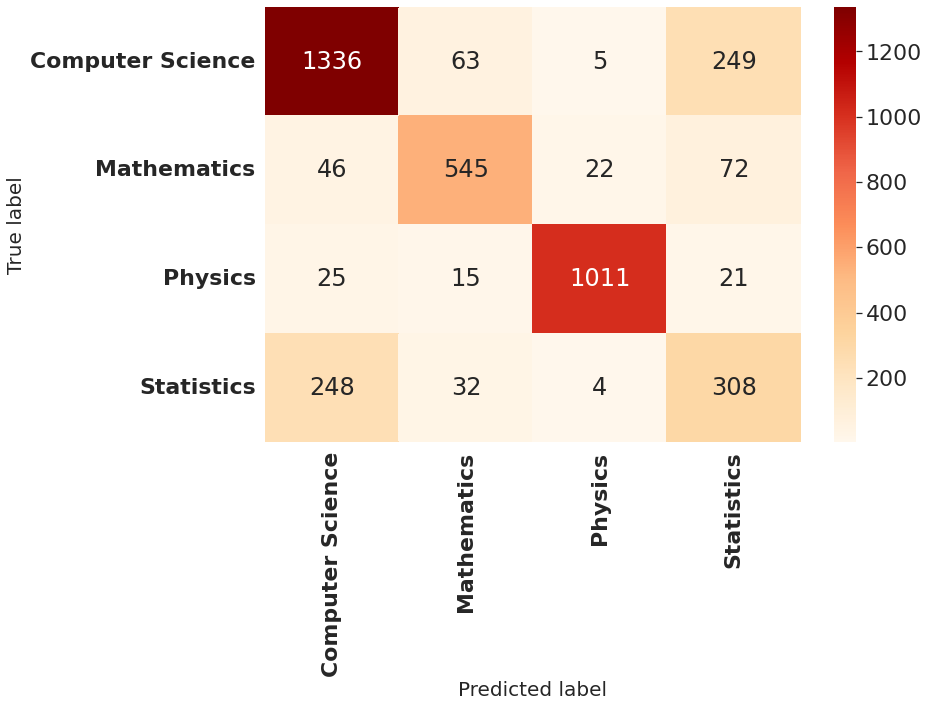
**Figure 4.2.1.1.1 Confusion matrix WordSeq+LSTM**

**Table 4.2.1** Class-wise classification report of WordSeq+LSTM

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 83.61 | 74.35 | 78.71 |
| **Mathematics** | 82.85 | 79.71 | 81.25 |
| **Physics** | 94.99 | 97.29 | 96.13 |
| **Statistics** | 45.55 | 59.63 | 51.65 |
| **Accuracy** | 79.24 | 79.24 | 79.24 |
| **Marco Avg** | 76.75 | 77.75 | 76.93 |
| **Weighted avg** | 80.90 | 79.24 | 79.81 |

**4.2.1.2 WordSeq + GRU**

Confusion matrix:



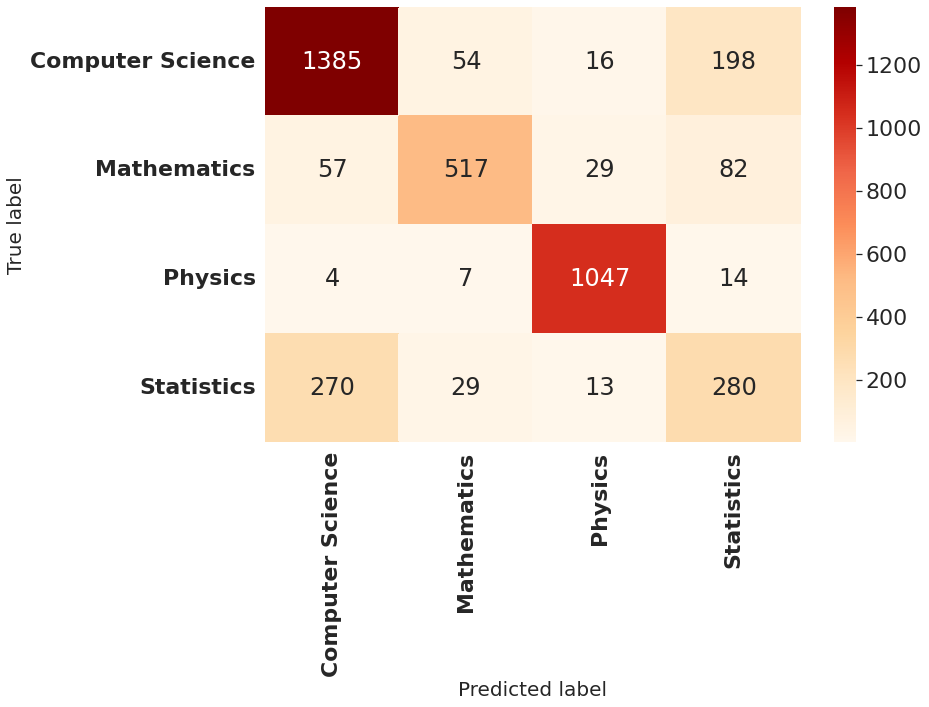
**Figure 4.2.1.2.1 Confusion matrix WordSeq+GRU**

**Table 4.2.2** Class-wise classification report of WordSeq+GRU

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 80.73 | 80.82 | 80.77 |
| **Mathematics** | 83.21 | 79.56 | 81.34 |
| **Physics** | 97.02 | 94.31 | 95.65 |
| **Statistics** | 47.38 | 52.03 | 49.60 |
| **Accuracy** | 79.96 | 79.96 | 79.96 |
| **Marco Avg** | 77.09 | 76.68 | 76.84 |
| **Weighted avg** | 80.58 | 76.96 | 80.24 |

**4.2.1.3 WordSeq + Bi-LSTM**

Confusion matrix:



**Figure 4.2.1.3.1 Confusion matrix WordSeq+Bi-LSTM**

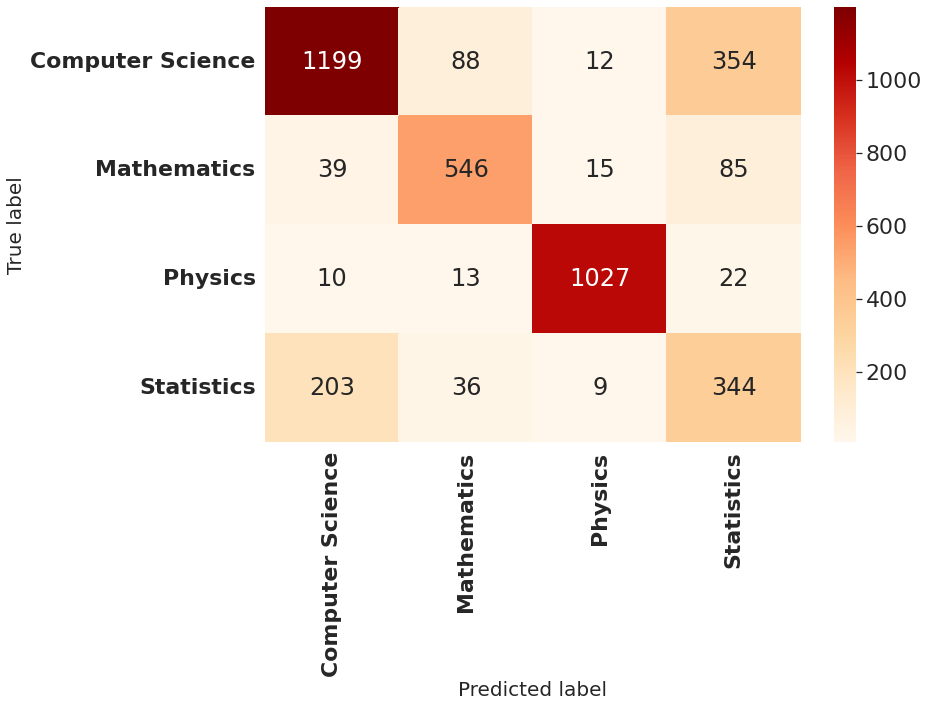
Using Bi LSTM, the same number of samples are predicted truly differently.Computer Science has 1385 true predictions whereas mathematics, physics and statistics have 517, 1047 and 280 truly predicted values

**Table 4.2.3** Class-wise classification report of WordSeq+Bi-LSTM

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 80.71 | 83.79 | 82.22 |
| **Mathematics** | 85.17 | 75.47 | 80.03 |
| **Physics** | 94.75 | 97.67 | 96.19 |
| **Statistics** | 48.78 | 47.30 | 48.03 |
| **Accuracy** | 80.68 | 80.68 | 80.68 |
| **Marco Avg** | 77.35 | 76.06 | 76.62 |
| **Weighted avg** | 80.51 | 80.68 | 80.53 |

**4.2.1.4 WordSeq + Bi-GRU**

Confusion matrix:



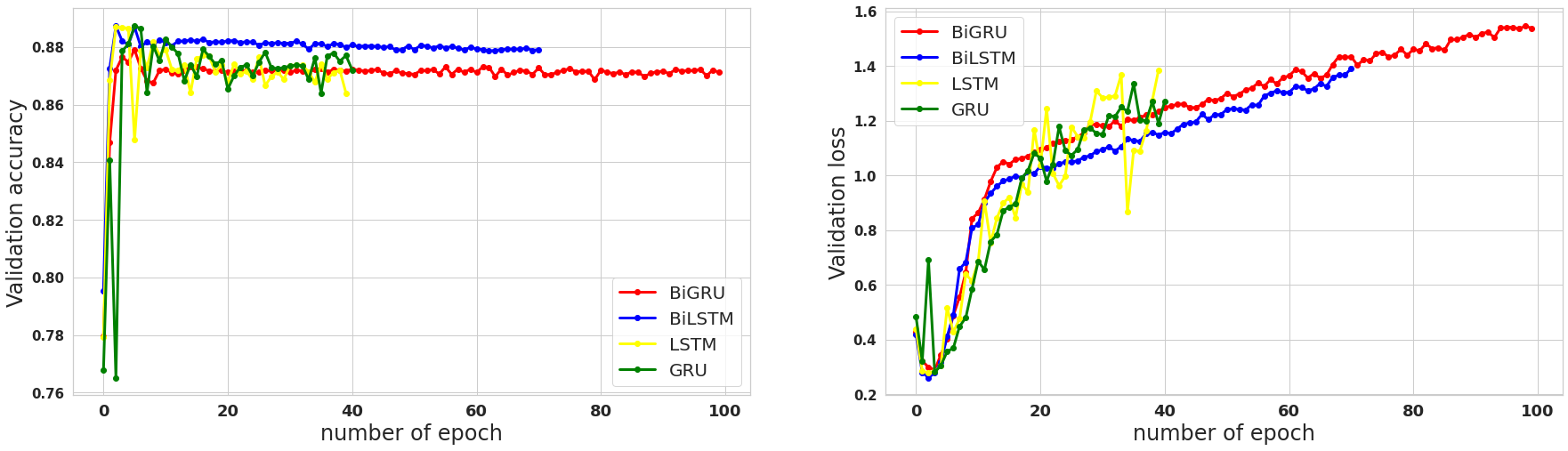
**Figure 4.2.1.3.1 Confusion matrix WordSeq+Bi-GRU**

According to the above-mentioned matrix, computer science class has 1653 instances in total among which 1199 samples are truly predicted which is the highest class wise result and in statistics the model predicted only 203 instances correctly.

**Table 4.2.3** Class-wise classification report of WordSeq+Bi-GRU

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 82.63 | 72.53 | 77.26 |
| **Mathematics** | 79.94 | 79.71 | 79.82 |
| **Physics** | 96.61 | 95.80 | 96.21 |
| **Statistics** | 42.73 | 58.11 | 49.25 |
| **Accuracy** | 77.86 | 77.86 | 77.86 |
| **Marco Avg** | 75.48 | 76.54 | 75.63 |
| **Weighted avg** | 80.01 | 77.86 | 78.63 |

**Comparing all the models of Word Sequence:**



**Figure 4.2.1.3.2** Overall validation Accuracy and validation loss in Word Sequence

In the validation dataset Bi-LSTM model is more stable and loss is minimum among others.

**Table 4.2.4** Overall performance result of Word Sequence

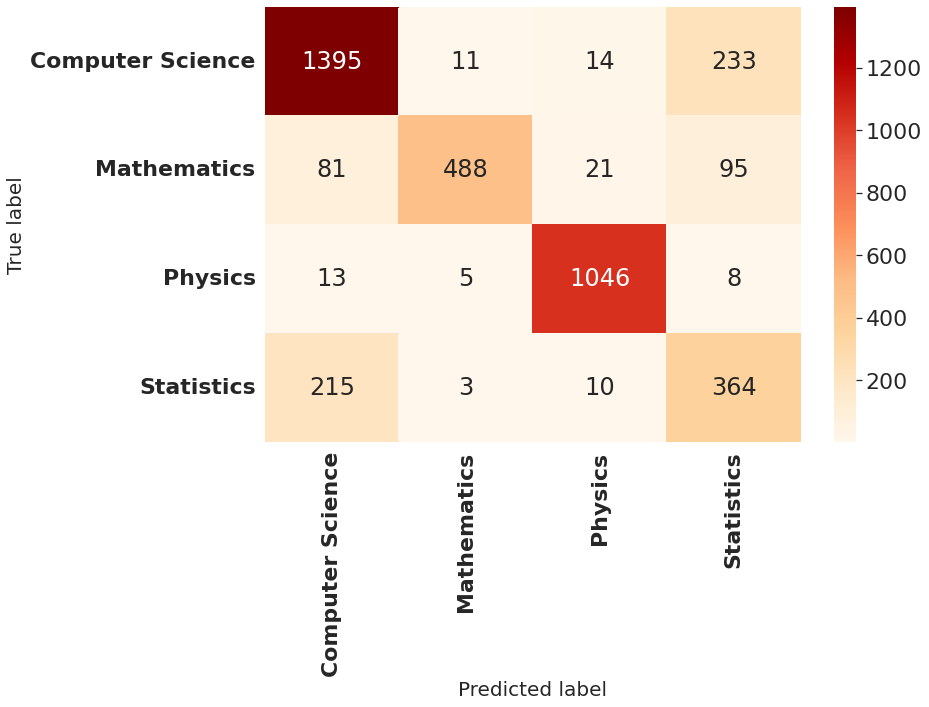
| **DL Algo.** | **Loss** | **Accuracy** | **Precision** | **Recall** | **F1 score** |
| --- | --- | --- | --- | --- | --- |
| **Bi GRU** | 0.36 | 0.88 | 0.86 | 0.91 | 0.89 |
| **Bi LSTM** | 0.25 | 0.89 | 0.88 | 0.92 | 0.90 |
| **GRU** | 0.34 | 0.89 | 0.89 | 0.92 | 0.90 |
| **LSTM** | 0.26 | 0.88 | 0.90 | 0.92 | 0.91 |

In terms of accuracy predicted from test data WordSeq+GRU gives the highest accuracy of 89.5% but in terms of loss, precision, recall and F1 score Bidirectional LSTM performs the best while using Word Sequence feature extraction.

**4.2.2 Word2Vec Approaches**

**4.2.2.1 Word2Vec + LSTM**

Confusion Matrix:



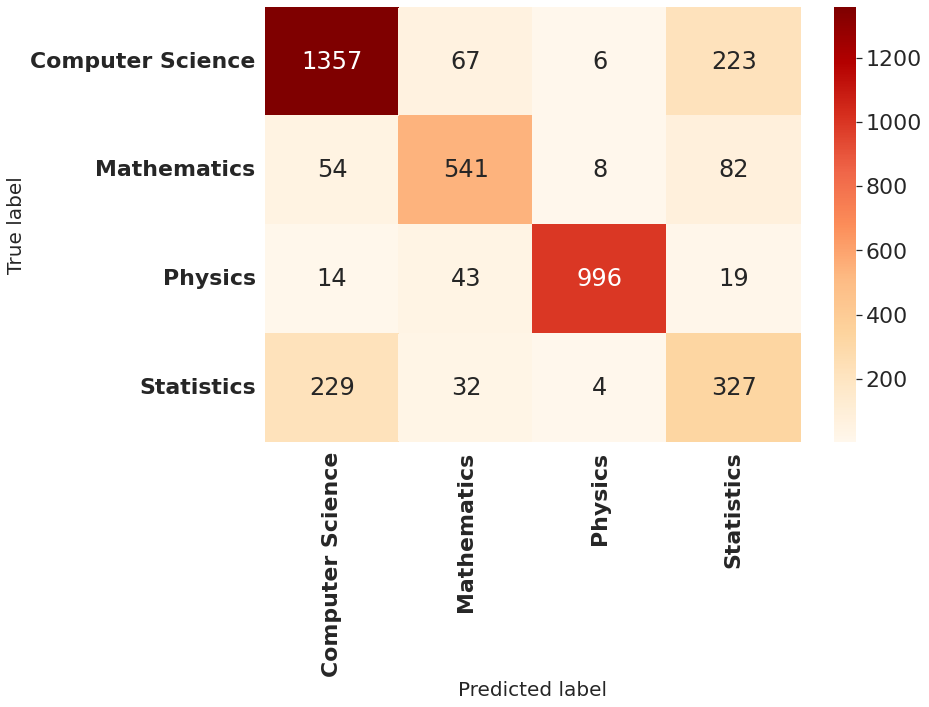
**Figure 4.2.2.1.1 Confusion matrix Word2Vec+LSTM**

**Table 4.2.5** Class-wise Classification report of Word2Vec+LSTM

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 81.87 | 84.39 | 83.11 |
| **Mathematics** | 96.25 | 71.24 | 81.88 |
| **Physics** | 95.88 | 97.57 | 96.72 |
| **Statistics** | 52.00 | 61.49 | 56.35 |
| **Accuracy** | 82.28 | 82.28 | 82.28 |
| **Marco Avg** | 81.50 | 76.67 | 79.51 |
| **Weighted avg** | 83.66 | 82.28 | 82.59 |

**4.2.2.2 Word2Vec + GRU**

Confusion Matrix:



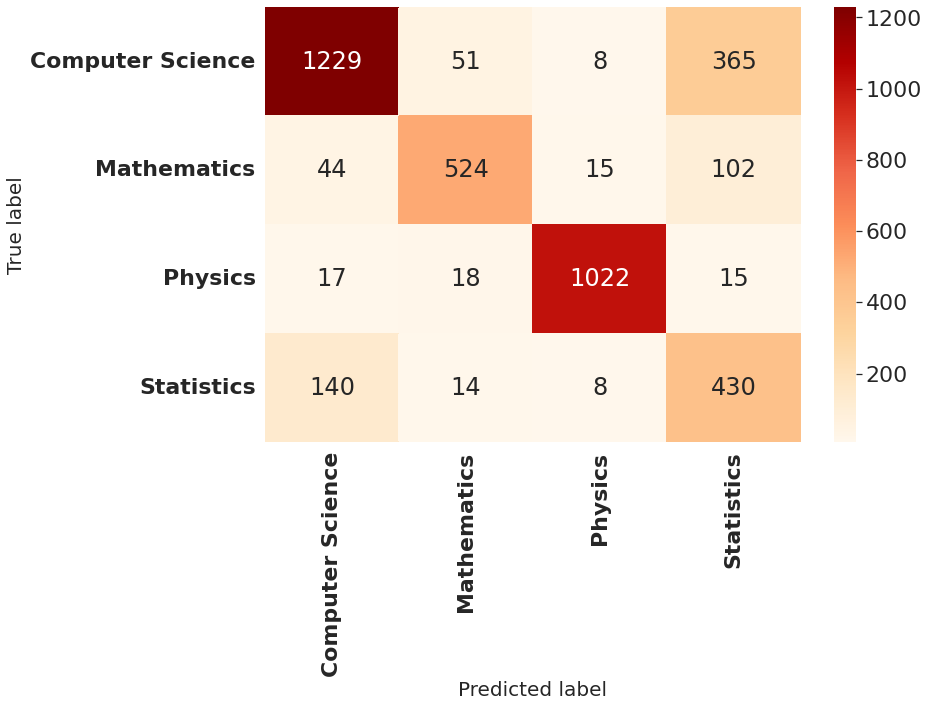
**Figure 4.2.2.2.1 Confusion matrix Word2Vec+GRU**

**Table 4.2.5** Class-wise Classification report of Word2Vec+GRU

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 82.04 | 82.09 | 82.07 |
| **Mathematics** | 79.21 | 78.98 | 79.09 |
| **Physics** | 98.22 | 92.91 | 95.49 |
| **Statistics** | 50.23 | 55.24 | 52.61 |
| **Accuracy** | 80.48 | 80.48 | 80.48 |
| **Marco Avg** | 77.43 | 77.30 | 77.32 |
| **Weighted avg** | 81.19 | 80.48 | 80.80 |

**4.2.2.3 Word2Vec + Bi-LSTM**

Confusion Matrix:



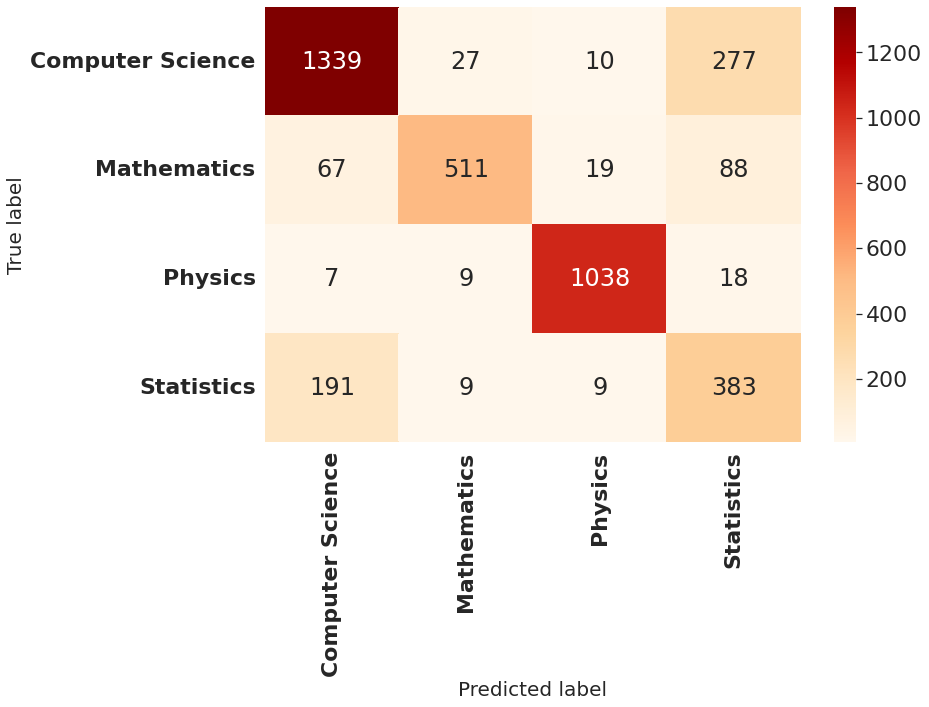
**Figure 4.2.2.3.1 Confusion matrix Word2Vec+Bi-LSTM**

**Table 4.2.6** Class-wise Classification report of Word2Vec+Bi-LSTM

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 85.94 | 74.35 | 79.73 |
| **Mathematics** | 86.33 | 76.50 | 81.11 |
| **Physics** | 97.06 | 95.34 | 96.19 |
| **Statistics** | 47.15 | 72.64 | 57.18 |
| **Accuracy** | 80.08 | 80.08 | 80.08 |
| **Marco Avg** | 79.12 | 79.70 | 78.55 |
| **Weighted avg** | 83.25 | 80.08 | 81.04 |

**4.2.2.4 Word2Vec + Bi-GRU**

Confusion Matrix:

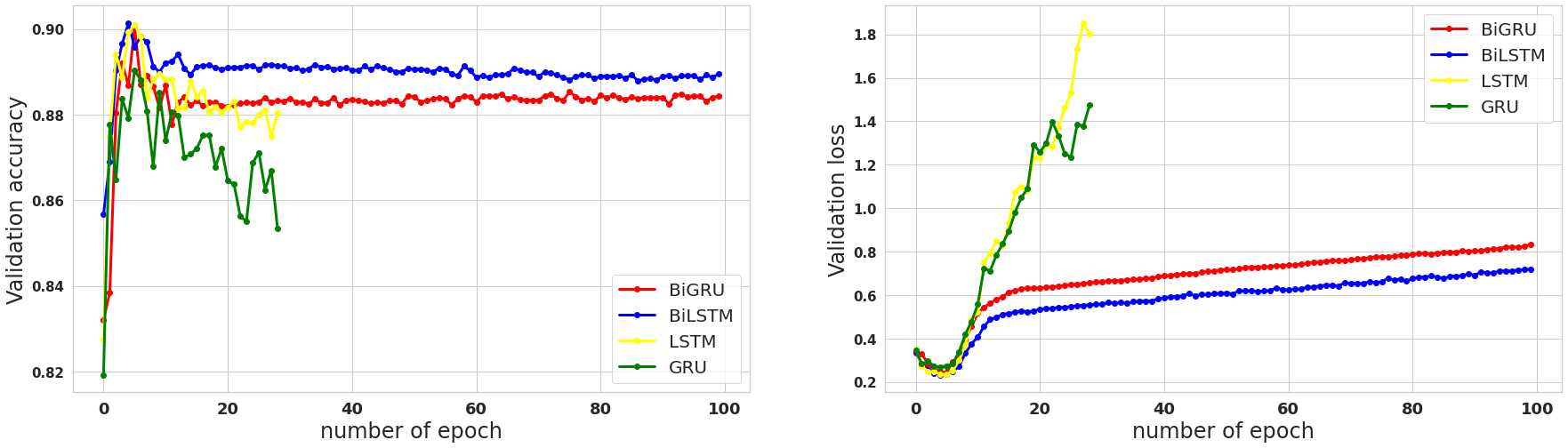


**Figure 4.2.2.4.1 Confusion matrix Word2Vec+Bi-GRU**

**Table 4.2.7** Class-wise Classification report of Word2Vec+Bi-GRU

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 83.48 | 81.00 | 82.22 |
| **Mathematics** | 91.91 | 74.60 | 82.35 |
| **Physics** | 96.47 | 96.83 | 96.65 |
| **Statistics** | 50.00 | 64.70 | 56.41 |
| **Accuracy** | 81.73 | 81.73 | 81.73 |
| **Marco Avg** | 80.46 | 79.28 | 79.41 |
| **Weighted avg** | 83.45 | 81.73 | 82.29 |

**Comparing all the models of Word2Vec:**

****

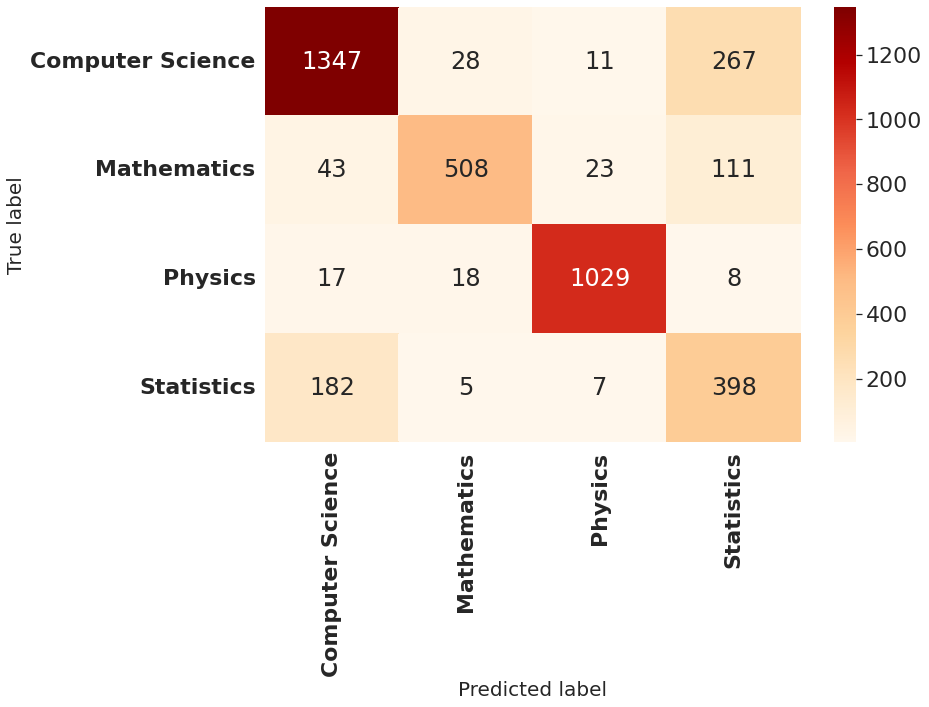
**Figure 4.2.2.4.2** Overall validation Accuracy and validation loss in Word2Vec

**Table 4.2.8** Overall performance result of Word2Vec

| **DL Algo** | **Loss** | **Accuracy** | **Precision** | **Recall** | **F1 score** |
| --- | --- | --- | --- | --- | --- |
| **Bi GRU** | 0.22 | 0.90 | 0.92 | 0.93 | 0.93 |
| **Bi LSTM** | 0.21 | 0.90 | 0.92 | 0.93 | 0.93 |
| **LSTM** | 0.22 | 0.90 | 0.92 | 0.93 | 0.93 |
| **GRU** | 0.24 | 0.89 | 0.91 | 0.92 | 0.91 |

**4.2.3 GloVe Approaches**

**4.2.3.1 GloVe + LSTM**

Confusion Matrix:

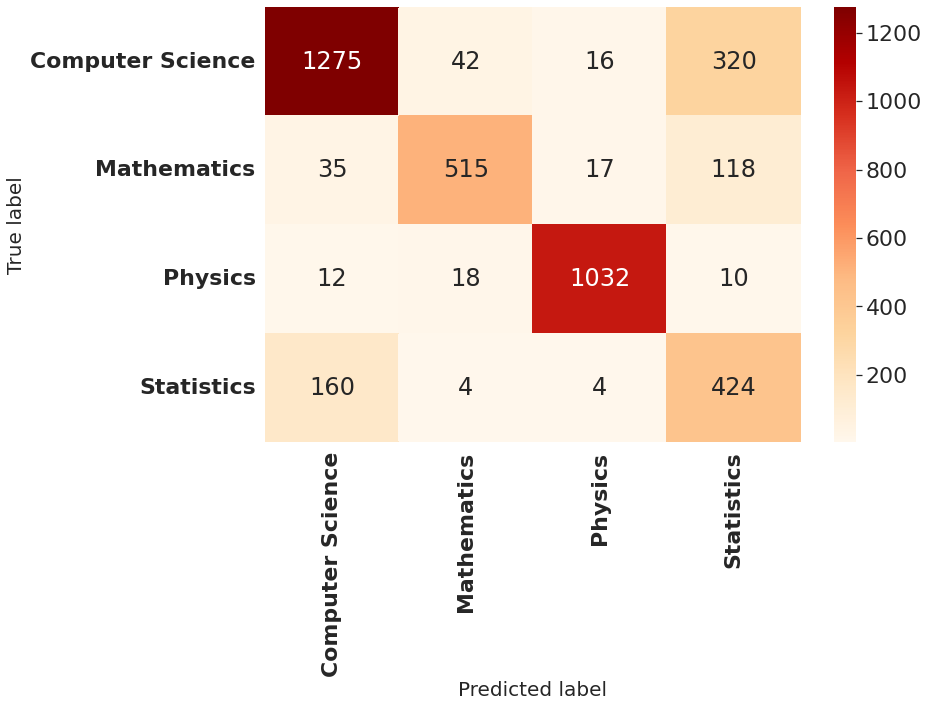
**Figure 4.2.3.1.1 Confusion matrix GloVe+LSTM**

**Table 4.2.9** Class-wise Classification report of GloVe+LSTM

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 84.77 | 81.49 | 83.10 |
| **Mathematics** | 90.88 | 74.16 | 81.67 |
| **Physics** | 96.17 | 95.99 | 96.08 |
| **Statistics** | 50.77 | 67.23 | 57.85 |
| **Accuracy** | 82.01 | 82.01 | 82.01 |
| **Marco Avg** | 80.65 | 79.72 | 79.67 |
| **Weighted avg** | 83.84 | 82.01 | 82.60 |

**4.2.3.2 GloVe + GRU**

Confusion Matrix:



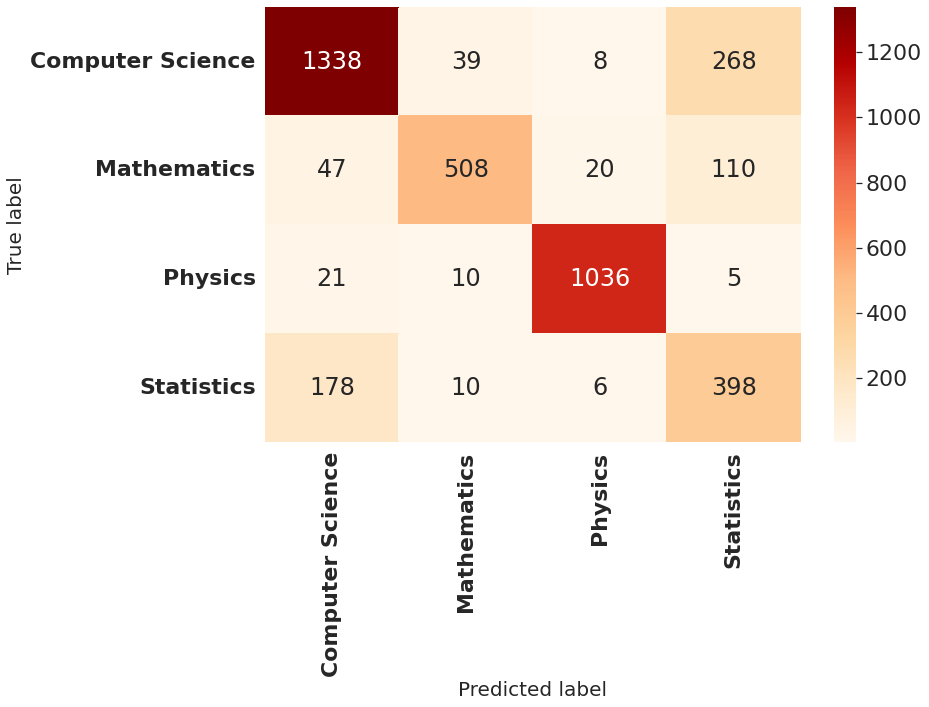
**Figure 4.2.3.2.1 Confusion matrix GloVe+GRU**

**Table 4.2.10** Class-wise Classification report of GloVe+GRU

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 86.03 | 77.13 | 81.34 |
| **Mathematics** | 88.95 | 75.18 | 81.49 |
| **Physics** | 96.54 | 96.27 | 96.40 |
| **Statistics** | 48.62 | 71.62 | 57.92 |
| **Accuracy** | 81.11 | 81.11 | 81.11 |
| **Marco Avg** | 80.04 | 80.05 | 79.29 |
| **Weighted avg** | 83.81 | 81.11 | 81.94 |

**4.2.3.3 GloVe + Bi-LSTM**

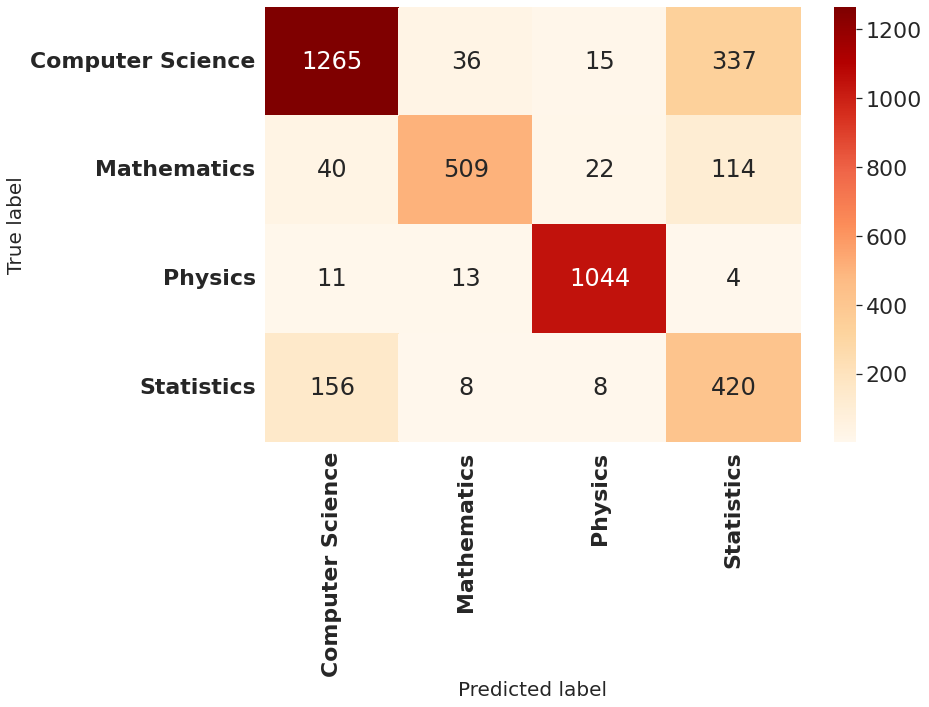
Confusion Matrix:

****

**Figure 4.2.3.3.1 Confusion matrix GloVe+Bi-LSTM**

**Table 4.2.11** Class-wise Classification report of GloVe+Bi-LSTM

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 84.47 | 80.54 | 82.67 |
| **Mathematics** | 89.59 | 74.16 | 81.15 |
| **Physics** | 96.82 | 96.64 | 96.73 |
| **Statistics** | 50.96 | 67.23 | 57.98 |
| **Accuracy** | 81.96 | 81.96 | 81.96 |
| **Marco Avg** | 80.46 | 79.74 | 76.63 |
| **Weighted avg** | 83.70 | 81.96 | 82.52 |

**4.2.3.4 GloVe + Bi-GRU**

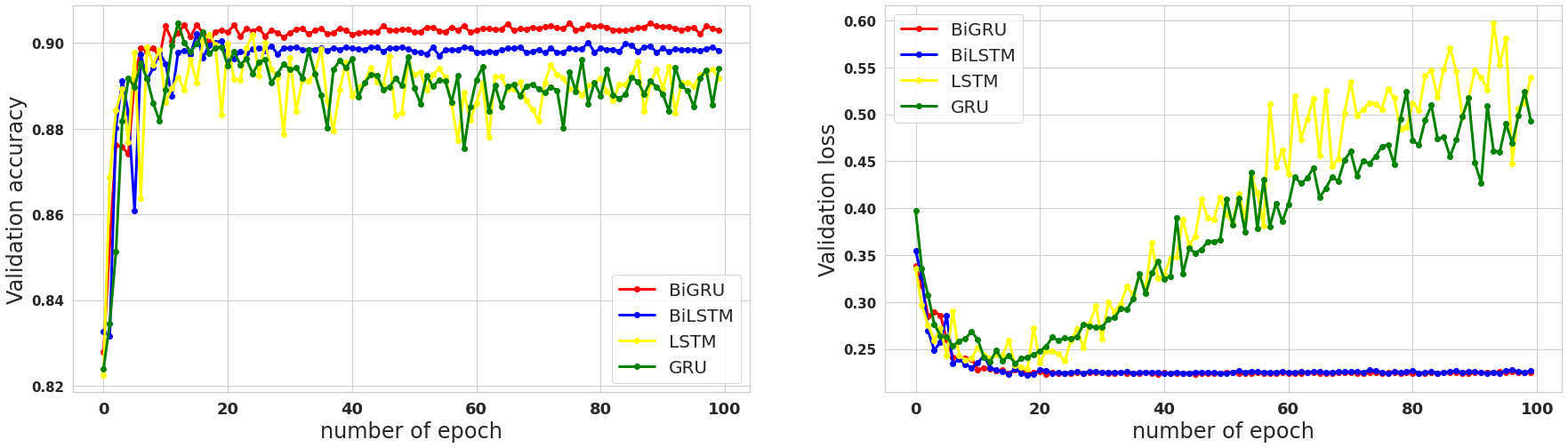
Confusion Matrix:

**Figure 4.2.3.4.1 Confusion matrix GloVe+Bi-GRU**

**Table 4.2.12** Class-wise Classification report of GloVe+Bi-GRU

| **Class/Label** | **Precision** | **Recall** | **F1-score** |
| --- | --- | --- | --- |
| **Computer Science** | 85.94 | 76.53 | 80.96 |
| **Mathematics** | 89.93 | 74.31 | 81.37 |
| **Physics** | 95.87 | 97.39 | 96.62 |
| **Statistics** | 48.00 | 70.95 | 57.26 |
| **Accuracy** | 80.91 | 80.91 | 80.91 |
| **Marco Avg** | 79.93 | 79.79 | 79.05 |
| **Weighted avg** | 83.67 | 80.91 | 81.72 |

**Comparing all the models of Glove:**

****

**Figure 4.2.3.4.2** Overall validation Accuracy and validation loss in GloVe

**Table 4.2.13** Overall performance result of GloVe

| **DL Algo** | **Accuracy** | **Loss** | **Precision** | **Recall** | **F1 Score** |
| --- | --- | --- | --- | --- | --- |
| **Bi LSTM** | 0.90 | 0.21 | 0.92 | 0.93 | 0.93 |
| **Bi GRU** | 0.90 | 0.21 | 0.92 | 0.94 | 0.93 |
| **LSTM** | 0.90 | 0.21 | 0.92 | 0.94 | 0.93 |
| **GRU** | 0.90 | 0.22 | 0.92 | 0.94 | 0.93 |

Confusion matrix is basically a parameter to measure the performance of a classification model from a given particular dataset and holds a 2D array in TensorFlow in python.

Here we did a comparative analysis among different procedures constituting confusion matrix to dignify the parameters like accuracy, recall, precisions etc.

By comparing all the models, all three approaches of feature extraction NLP techniques we can decide or come to a point where and which deep learning approaches performed better. And as study suggests that in the experiment of 4 different deep learning algorithms with 3 different feature extraction processes we can present better approaches for extracting features on textual dataset. We implemented time-sequence algorithms to conduct the experiment in order to draw out a comparison on the process of feature extraction, as from the table of accuracy, lost and precision we can see we implemented LSTM, Bi-LSTM, GRU, Bi-GRU in all 3 feature extraction process form these table we can decide or come to a conclusion that Glove works better in vectorization process on textual data set rather than word2vec and word sequence. Glove is a pre-train data set on a corpus which has an estimated 6 billion datasets which concludes the matter with better accuracy and less loss.

GloVe is an unsupervised learning algorithm for getting vector representations of words. Training is performed using aggregated global words and word co-occurrence statistics from the corpus, and the resulting plot reveals an interesting linear substructure of the word vector space. Word2Vec uses text as training data for neural networks. The resulting embedding captures whether the word occurs in a similar context. GloVe focuses on the appearance of common words throughout the corpus. The embedding indicates the probability that two words will occur at the same time. GloVe showed how to use the global statistics contained in the document. FastText is built on the word2vec model, but instead of considering words, let's look at subwords. From the tables, with suitable reasons we can decide why gloves work better than feature extraction.

Let us discuss the table of accuracy where Bi-LSTM , LSTM works better with GloVe than any other deep learning techniques and approaches but for word sequence bi-gru performed poorly than any other algorithms .Basically, Bi-LSTM is giving the best F1 score and accuracy and LSTM and Bi-GRU is working good as well.

For example, in the case of Bi-GRU the precision is 0.92 which indicates the sample holding this probability is much premuseable to be positive and precise than the samples in the positive class. Similarly, recall is 0.93 which means the recall is very close to 1 and a model having high precision and recall score always produces output sensitive predictions and performs with better results.

As we can see, Bi-LSTM, LSTM, GRU are also having high precision and recall values so it works really well and almost everything is very close to 1.

Regarding the F1-score, Bi-LSTM has 0.93 which is slightly more than other approaches. It means better performance as F1 score takes both false positives and false negatives both into account. It declares that Bi-LSTM is working better as in this case accuracy measurement is not better than F1-score measurement. F1-score gives the best measurement in this model.

For both LSTM and GRU the validation loss is at an increasing rate so that means the model will not work well for validation data but work well for training data which is an overfitting situation. Hence, Bi-GRU and Bi LSTM work well in this case.

When we set the average parameter to macro for f1-score, the function returns the average without considering the ratio of each label. The average accuracy of the macro is 0.5 and the weighted average is 0.7. The weighted average of this model is high because the location of the inaccuracy was better, but it is underestimated in this dataset (1/5 only) and is not considered much in the weighted average.

Chapter 5

# Conclusion and Future Works

For the task of text classification, deep learning classifiers have been used on different domains such as restaurants, business industries etc. However there is a lack of research on the topic of research article classification. Due to this problem readers are encountering difficulties finding out proper keywords of a paper. Automatic detection of a particular keyword of an article can help writers to reach more readers and with the use of deep learning techniques, we can successfully solve the problem. Prior studies have proven that the task can be achieved with a significant accuracy score but still there is room for improvement. In this study, we presented a supervised Machine learning and deep learning approach to classify research papers into four categories. The goal of our research was to automate the system of identifying research articles using deep learning algorithms trained on three different word embedding techniques. The result indicated that among the baseline models, random forest obtained the highest accuracy of 84%. With the use of deep learning classifiers, we improved the performance of baseline models where the Bi-LSTM model trained on GloVe embedding performed with the highest accuracy score of 90% and f1-score of 93%. The reason behind the superior performance of GloVe embedding was that GloVe focuses on words co-occurrences over the whole dataset. Furthermore, Bi-GRU trained on word to sequence embedding performed the worst among all classifiers with an accuracy and f1-score of 88% and 89%, respectively. It is worth mentioning that the statistics class had the most wrong predictions which highly influenced the classification scores of all the classifiers. The results can be further improved if we can gather more articles related to statistics class and train them for a longer time.

However, our study faced some limitations such as small number of training data and less number of classes. In addition, we did not build any real world web application that can identify the classes based on the abstract of an article. In future, we want to develop a real world application that can help researchers find keywords of a paper on a model that is trained on more sophisticated transfer learning algorithms like BERT and Ro-BERT.

# References

[1] Kim, Sang-Woon, and Joon-Min Gil. "Research paper classification systems based on TF-IDF and LDA schemes." *Human-centric Computing and Information Sciences* 9.1 (2019): 1-21.

[2] Moriya, Shun, and Chihiro Shibata. "Transfer learning method for very deep CNN for text classification and methods for its evaluation." *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*. Vol. 2. IEEE, 2018.

[2] Al Amrani, Yassine, Mohamed Lazaar, and Kamal Eddine El Kadiri. "Random forest and support vector machine based hybrid approach to sentiment analysis." *Procedia Computer Science* 127 (2018): 511-520.

[3] Hertzmann, Aaron, David Fleet, and Marcus Brubaker. "Machine learning and data mining lecture notes." *Computer Science Department, University of Toronto* (2010).

[4] Burman, Iti, and Subhranil Som. "Predicting students academic performance using support vector machine." *2019 Amity international conference on artificial intelligence (AICAI)*. IEEE, 2019.

[5] Liu, Qiuge, Qing He, and Zhongzhi Shi. "Extreme support vector machine classifier." *Pacific-asia conference on knowledge discovery and data mining*. Springer, Berlin, Heidelberg, 2008.

[6] Chakraborty, Arijit, et al. "Determining Protein–Protein Interaction Using Support Vector Machine: A Review." *IEEE Access* 9 (2021): 12473-12490.

[7] Yu, Lei. "Information Extraction and Classification on Journal Papers." (2021).

[8] Pal, Mahesh. "Random forest classifier for remote sensing classification." *International journal of remote sensing* 26.1 (2005): 217-222.

[9] De Mántaras, R. López. "A distance-based attribute selection measure for decision tree induction." *Machine learning* 6.1 (1991): 81-92.

[10] Farris, Frank A. "The Gini index and measures of inequality." *The American Mathematical Monthly* 117.10 (2010): 851-864.

[11] Algehyne, Ebrahem A., et al. ``Fuzzy neural network expert system with an improved Gini index random forest-based feature importance measure algorithm for early diagnosis of breast cancer in Saudi Arabia." *Big Data and Cognitive Computing* 6.1 (2022): 13.

[12] Quinlan, J. Ross. "Program for machine learning." *C4. 5* (1993).

[13] Pal, Mahesh. "Random forest classifier for remote sensing classification." *International journal of remote sensing* 26.1 (2005): 217-222.

[14] Breiman, Leo. "1 RANDOM FORESTS--RANDOM FEATURES." (1999).

[15] Xu, Haotian, et al. "Text classification with topic-based word embedding and convolutional neural networks." *Proceedings of the 7th ACM International Conference on Bioinformatics, Computational Biology, and Health Informatics*. 2016.

[16] Stein, Roger Alan, Patricia A. Jaques, and Joao Francisco Valiati. "An analysis of hierarchical text classification using word embeddings." *Information Sciences* 471 (2019): 216-232.

[17] Singh, Ksh Nareshkumar, et al. "A novel approach for dimension reduction using word embedding: An enhanced text classification approach." *International Journal of Information Management Data Insights* 2.1 (2022): 100061.

[18] Ge, Lihao, and Teng-Sheng Moh. "Improving text classification with word embedding." *2017 IEEE International Conference on Big Data (Big Data)*. IEEE, 2017.

[19] Mikolov, Tomas, et al. "Efficient estimation of word representations in vector space." *arXiv preprint arXiv:1301.3781* (2013).

[20] Minaee, Shervin, et al. "Deep learning--based text classification: a comprehensive review." *ACM Computing Surveys (CSUR)* 54.3 (2021): 1-40.

[21] Sherstinsky, Alex. "Fundamentals of recurrent neural network (RNN) and long short-term memory (LSTM) network." *Physica D: Nonlinear Phenomena* 404 (2020): 132306.

[22] Sun, Lei, et al. "Multiple-target deep learning for LSTM-RNN based speech enhancement." *2017 Hands-free Speech Communications and Microphone Arrays (HSCMA)*. IEEE, 2017.

[23] Zulqarnain, Muhammad, et al. "Text classification based on gated recurrent unit combines with support vector machine." *International Journal of Electrical and Computer Engineering* 10.4 (2020): 3734.

[24] Van Huynh, Tin, et al. "Hate speech detection on vietnamese social media text using the bi-gru-lstm-cnn model." *arXiv preprint arXiv:1911.03644* (2019).

[25] Gruber, Nicole, and Alfred Jockisch. "Are GRU cells more specific and LSTM cells more sensitive in motive classification of text?." *Frontiers in artificial intelligence* 3 (2020): 40.

[26] Tang, Qinting, et al. "Full attention-based bi-GRU neural network for news text classification." *2019 IEEE 5th International Conference on Computer and Communications (ICCC)*. IEEE, 2019.

[27] Cho, Kyunghyun, et al. "Learning phrase representations using RNN encoder-decoder for statistical machine translation." *arXiv preprint arXiv:1406.1078* (2014).

[28] Zulqarnain, Muhammad, et al. "An improved deep learning approach based on variant two-state gated recurrent unit and word embeddings for sentiment classification." *International Journal of Advanced Computer Science and Applications* 11.1 (2020).

[29] Forman, George. "An extensive empirical study of feature selection metrics for text classification." *J. Mach. Learn. Res.* 3.Mar (2003): 1289-1305.

[30] Kim, Sang-Woon, and Joon-Min Gil. "Research paper classification systems based on TF-IDF and LDA schemes." *Human-centric Computing and Information Sciences* 9.1 (2019): 1-21.

[31] Minaee, Shervin, et al. "Deep learning--based text classification: a comprehensive review." *ACM Computing Surveys (CSUR)* 54.3 (2021): 1-40.

[32] Liu, Jingzhou, et al. "Deep learning for extreme multi-label text classification." *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*. 2017.

[33] Li, Yue, Xutao Wang, and Pengjian Xu. "Chinese text classification model based on deep learning." *Future Internet* 10.11 (2018): 113.

[34] Ali, Farman, et al. "Transportation sentiment analysis using word embedding and ontology-based topic modeling." *Knowledge-Based Systems* 174 (2019): 27-42.

[35] DHARMA, EDDY MUNTINA, et al. "The Accuracy Comparison Among Word2vec, Glove, And Fasttext Towards Convolution Neural Network (CNN) Text Classification." *Journal of Theoretical and Applied Information Technology* 100.2 (2022).

[36] Ria, Nushrat Jahan, et al. "Toward an enhanced bengali text classification using saint and common form." *2020 11th international conference on computing, communication and networking technologies (ICCCNT)*. IEEE, 2020.

[37] Wei, Zhihua, et al. "Feature selection on Chinese text classification using character n-grams." *International conference on rough sets and knowledge technology*. Springer, Berlin, Heidelberg, 2008.

[38] Prasetijo, Agung B., et al. "Hoax detection system on Indonesian news sites based on text classification using SVM and SGD." *2017 4th International Conference on Information Technology, Computer, and Electrical Engineering (ICITACEE)*. IEEE, 2017.

[39] Lakhotia, Suyash, and Xavier Bresson. "An experimental comparison of text classification techniques." *2018 International Conference on Cyberworlds (CW)*. IEEE, 2018.

[40] Rabbimov, I. M., and S. S. Kobilov. "Multi-class text classification of Uzbek news articles using machine learning." *Journal of Physics: Conference Series*. Vol. 1546. No. 1. IOP Publishing, 2020.

[41] Das, Supriya, et al. "Hand-written and machine-printed text classification in architecture, engineering & construction documents." *2018 16th International Conference on Frontiers in Handwriting Recognition (ICFHR)*. IEEE, 2018.

[42] Kandimalla, Bharath, et al. "Large scale subject category classification of scholarly papers with deep attentive neural networks." *Frontiers in research metrics and analytics* 5 (2021): 600382.

[43] Yue, Tan, et al. "PaperNet: A Dataset and Benchmark for Fine-Grained Paper Classification." Applied Sciences 12.9 (2022): 4554.

[44] Ech-Chouyyekh, Monir, Hicham Omara, and Mohamed Lazaar. "Scientific paper classification using Convolutional Neural Networks." Proceedings of the 4th International Conference on Big Data and Internet of Things. 2019.

[45] Ali, Farman, et al. "Transportation sentiment analysis using word embedding and ontology-based topic modeling." *Knowledge-Based Systems* 174 (2019): 27-42.

[46] Suneera, C. M., and Jay Prakash. "Performance analysis of machine learning and deep learning models for text classification." *2020 IEEE 17th India Council International Conference (INDICON)*. IEEE, 2020.

[47] Liu, Yuanchao, et al. "Abstract sentence classification for scientific papers based on transductive SVM." *Computer and Information Science* 6.4 (2013): 125.

[48] Chowdhury, Shovan, and Marco P. Schoen. "Research paper classification using supervised machine learning techniques." *2020 Intermountain Engineering, Technology and Computing (IETC)*. IEEE, 2020.

[49] Kim, Sang-Woon, and Joon-Min Gil. "Research paper classification systems based on TF-IDF and LDA schemes." *Human-centric Computing and Information Sciences* 9.1 (2019): 1-21.

[50] Dien, Tran Thanh, Bui Huu Loc, and Nguyen Thai-Nghe. "Article classification using natural language processing and machine learning." *2019 International Conference on Advanced Computing and Applications (ACOMP)*. IEEE, 2019.

[51] Jaya, I., et al. "Scientific documents classification using support vector machine algorithm." *Journal of Physics: Conference Series*. Vol. 1235. No. 1. IOP Publishing, 2019.

[52] Mustafa, Ghulam, et al. "Multi-label classification of research articles using Word2Vec and identification of similarity threshold." *Scientific Reports* 11.1 (2021): 1-20.

[53] Jarvis, Scott, and Scott A. Crossley, eds. *Approaching Language Transfer Through Text Classification: Explorations in the Detection based Approach*. Vol. 64. Multilingual Matters, 2012.