DEEP LEARNING METHODS WITH PRE-TRAINED WORD EMBEDDINGS AND PRE-TRAINED TRANSFORMERS FOR EXTREME MULTI LABEL TEXT CLASSIFICATION

NECDET EREN ERCİYES

FEBRUARY 2022

ÇANKAYA UNIVERSITY

GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES

DEPARTMENT OF COMPUTER ENGINEERING

MASTER'S THESIS IN

COMPUTER ENGINEERING

DEEP LEARNING METHODS WITH PRE-TRAINED WORD EMBEDDINGS AND PRE-TRAINED TRANSFORMERS FOR EXTREME MULTI LABEL TEXT CLASSIFICATION

NECDET EREN ERCİYES

FEBRUARY 2022

ABSTRACT

DEEP LEARNING METHODS WITH PRE-TRAINED WORD EMBEDDINGS AND PRE-TRAINED TRANSFORMERS FOR EXTREME MULTI LABEL TEXT CLASSIFICATION

ERCİYES, Necdet Eren

Master of Science in Computer Engineering

Supervisor: Assoc. Prof. Dr. Abdül Kadir GÖRÜR

February 2022, 49 pages

In recent years, there appears to be a remarkable increase int the number of textual documents online. This increase requires the creation of highly improved machine learning methods to classify text in many different domains. The effectiveness of these machine learning methods depends on the model capacity to understand the complex nature of the unstructured data and the relations of the features that exist. Many different machine learning methods were proposed for a long time to solve text classification problems, such as SVM, kNN and Rocchio classification. These shallow learning methods have achieved doubtless success in many different domains. For big and unstructured data like text, deep learning methods which can learn representations and features from the input data without using any feature extraction methods have shown to be one of the major solutions. In this study, we explore the accuracy of recent recommended deep learning methods for multi-label text classification starting with simple RNN, CNN models to pre-trained transformer models. We evaluated these methods' performances by computing multi-label evaluation metrics and compared the results with the previous studies.

Keywords: Multi-Label Text Classification, Machine Learning, Deep Learning, Word Embedding, Transformers

ÖZ

ÇOKLU ETİKET SINIFLANDIRMASI İÇİN ÖNCEDEN EĞİTİLMİŞ KELİME VEKTÖRLERİ VE ÖNCEDEN EĞİTİLMİŞ TRANSFORMER MODELLERİ İLE DERİN ÖĞRENME YÖNTEMLERİ

ERCİYES, Necdet Eren
Bilgisayar Mühendisliği Yüksek Lisans

Danışman: Dr. Öğr Üyesi Abdül Kadir GÖRÜR
Şubat 2022, 49 sayfa

Son yıllarda, çevrimiçi metin belgelerinde dikkat çekici sayısal bir artış olduğu görülmektedir. Bu artış, metni birçok farklı alanda sınıflandırmak için oldukça gelişmiş makine öğrenimi yöntemlerinin oluşturulmasını gerektiriyor. Bu makine öğrenimi yöntemlerinin etkinliği, yapılandırılmamış verilerin karmaşık doğasını ve var olan özelliklerin ilişkilerini anlamak için model kapasitesine bağlıdır. Metin sınıflandırma problemlerini çözmek için uzun süredir SVM, kNN, Rocchio sınıflandırması gibi birçok farklı makine öğrenmesi yöntemi önerilmiştir. Bu sığ öğrenme yöntemleri, pek çok farklı alanda şüphesiz başarıya ulaşmıştır. Metin gibi büyük ve yapılandırılmamış veriler için, herhangi bir özellik çıkarma yöntemi kullanmadan giriş verilerinden temsilleri ve özellikleri öğrenebilen derin öğrenme yöntemleri önemlil çözümlerden biri olarak gösterilmiştir. Bu çalışmada, basit RNN, CNN modellerinden önceden eğitilmiş transformer modellerine kadar çok etiketli metin sınıflandırması için son önerilen derin öğrenme yöntemlerinin doğruluğunu araştırıyoruz. Bu yöntemlerin performanslarını çok etiketli değerlendirme ölçütlerini hesaplayarak değerlendirdik ve sonuçları önceki çalışmalarla karşılaştırdık.

Anahtar Kelimeler: Çok Etiketli Metin Sınıflandırma, Makine Öğrenmesi, Derin Öğrenme, Kelime Yerleştirme, Transformerlar

ACKNOWLEDGEMENT

I would like to express my sincere gratitude to my parents for their support and sacrifice to me. Your memories would ever shine in my mind.

Special thanks to my supervisor Asst. Prof. Dr. Abdül Kadir GÖRÜR for the excellent guidance and providing me with an excellent atmosphere to conduct this research. My special gratitude also goes to the rest of the thesis committee members Asst. Prof. Dr. Uğur SOPAOĞLU and Asst. Prof. Dr. Erdal ERDAL for the encouragement and insightful comments

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF SYMBOLS AND ABBREVIATIONS

ABBREVIATIONS

NLP	Natural Language Processing
DL	Deep Learning
DNN	Deep Neural Network
ML	Machine Learning
XMTC	Extreme Multi-Label Classification
NN	Neural Network
CNN	Convolutional Neural Network
RNN	Recurrent Neural Network
LSTM	Long Short Term Memory
GRU	Gated Recurrent Unit
KNN	K-Nearest Neighbor
GPU	Graphics Processing Unit
TPU	:Tensor Processing Unit
HAN	:Hierarchical Attention Network

CHAPTER I

INTRODUCTION

1.1 MOTIVATIONS

Unstructured data that is stored as text can be found in many places: messages sent from social media accounts, web pages that have information presented with a user interface and academic surveys that store answers for the survey questions. Text is a type of data that is produced every day from different sources and it may contain very important information, but for this information to be usable, the process of revealing meaningful information from within the text may be necessary. This process could be very hard because the text is an instance of unstructured data. Today solving the text classification problem has become important due to the increasing number of online textual documents and the need that will require the analysis of these documents to be used for predictions in many different areas.

Text classification is applicable everyday in our lives: from news and articles categorized by content to email spam detection.

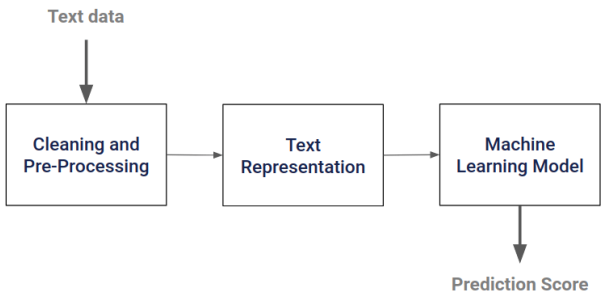

Figure 1.1: Text Classification Process

Natural language processing is a concept that emerged as a result of the use of artificial intelligence algorithms as a result of people's need to analyze and model the language using the processing power and speed of computers. Most NLP tools use machine learning methods to understand human language and these methods rely on

machine to analyze and represent human language numerically. For a long time, most of the methods used to study NLP problems employed shallow machine learning methods and hand drawn features which consumed a large part of the time that was reserved for. This lead to issues such as designing models for higher dimensional feature space with more errors and a higher run time complexity since linguistic information was represented with sparse representations. However, with the increase in models using the word embedding technique, which is considered succesful, neural based models have achieved excellent results on various language-related tasks as compared to traditional machine learning models like support vector machine (SVM) or logistic regression [1].

Deep learning, which is comparable to human brain in terms of having neurons that transform input data by changing weights and applying activation functions to give the data to the next layer, is the biggest and fastest growing technique in artificial intelligence. In this thesis, different deep learning methods are applied for text classification task to get comparable results with the state of the art.

In this study, we aim to use different deep learning methods on Reuters Corpus Volume 1-v2 data set to get text classification results comparable to state of the art. These methods include basic neural network architectures such as convolutional neural networks (CNN), recurrent neural networks (RNN) as well as advanced NLP techniques such as transformers.

1.2 WHAT IS DEEP LEARNING

It is a learning algorithm with several hidden layers that is studied under the category of machine learning algorithms which can increase its performance depending on the growth of the data. While the performance of traditional machine learning algorithms is not directly proportional to the volume of data, deep learning systems' performance increase in parallel with the size of the data.

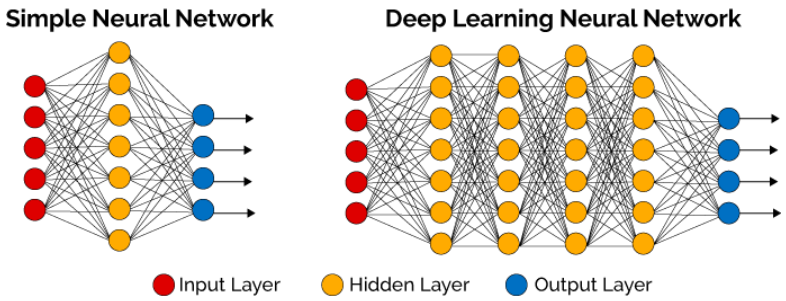

Figure 1.2: Deep Learning Neural Network

Deep learning neural networks have neurons. As can be seen in figure 2.1, the neurons are represented with circles. There are connections between these neurons that pass the data to the next neuron, which is located in the next layer, after the activation function is applied. These neurons are inter-connected. The neurons are grouped into three different types of layers: Input layer, hidden layers and output layer. The input layer receives input data, the hidden layers apply mathematical operations to optimize the weights and finally normalize the data with the activation function and the output layer returns the output data. Each connection between neurons has a weight that is multiplied with the input data. This weight implies the importance of the input feature. The initial weights of the deep learning model are generated randomly.

The primary aim of this study is to implement different deep learning methods and test these methods on Reuters Corpus Volume 1 data set. There are various reasons why we included deep learning methods in the experimental studies that are carried out for this work. The path to deep learning methods first started with the discovery of machine learning methods. Before machine learning methods, it was expected to enter predetermined rules for solving problems such as classification and regression, and the algorithm to produce results according to these rules. In the process of determining these rules, the opinions of experts in the specific filed or previously successful experimental studies were used for classification or regression problems.

Improvements were constantly being made to define better rules, which led to costly processes. With the advent of machine learning algorithms, rule-setting studies have shifted to find more data. This change was possible when machine learning algorithms could learn on their own by analyzing the patterns in the data. In order to better understand the contribution of deep learning methods to classification problems, we need to examine, in a simple way, how a data set containing news item documents

is classified by rule-based, machine learning and deep learning methods.

Let us understand how rule-based systems work in text classification process. First of all, we know that certain news categories are associated with certain keywords, and based on this information, we need to extract the keywords from the document containing the news item text. Then we need to determine which documents this news item text belongs to which class values using basic if else blocks. This method depends entirely on the success of rule-making methods and requires costly processes.

If we do the same classification process in a machine learning algorithm, our focus will be on the data rather than the rules. Firstly, in order for machine learning methods to work effectively, various features must be extracted from the data that contains news items. For the document with the news item text, we can keep a list of how many times each word is used in the textual document, which is called bag of words in the literature. We can also extract n gram features such as single, double and triple word sequences from the text and combine them with the bag of words vector. Part of speech tagging method can help us make a better classification by learning the relationship between the structure of the sentences and the assigned class values by revealing which speech tag used in the sentence. Therefore, we can combine the part of speech vector with our input vector that will be used by a machine learning algorithm. Then we can perform the classification of news item texts by giving the input data vector to logistic regression algorithm. The logistic regression algorithm will place the vector of each news item text in the training set in multidimensional space and determine the weights of the non-linear equation that will best separate each class value. In the search for the best weights, the actual class values are compared with the estimated class values and the weights and bias value are corrected so that the values are equal.

Finally, suppose that we perform classification using deep learning algorithm. When the deep learning algorithm is used, we will not need to use special feature extraction methods from the news item texts in the training data set, instead, it will be sufficient to convert the texts into vectors numerically. These vectors can be bag of words, tf-idf vectors based on whether the word occurs in the document, or word embedding vectors computed using deep learning methods. There are two important points that affect the accuracy of the deep learning algorithm results. The first point is

the size of the data and the second point is the type of the operation that is performed in the hidden layers such as convolution, sequential alignment or attention methods. Suppose a deep learning algorithm uses the attention mechanism as an example and classifies news item documents by giving weight to some words and phrases in the input data. To perform the classification process in the last layer, we should use as many nodes as the number of different classes and assign the values which we found in the range of 0-1 and higher than pre-determined threshold value to the corresponding class value. In conclusion if the training data set is large enough and the attention mechanism is suitable for the data structure, the deep learning method is more likely to provide better accuracy rates than other machine learning methods and rule based methods.

1.3 ORGANIZATION OF THE THESIS

This thesis contains six chapters.

In Chapter 2 related work on RCV1 data set is briefly explained.

Chapter 3 includes an introduction to different text classification problems.

In Chapter 3, different text classification approaches for different domains are explained such as binary classification and multi class classification.

In Chapter 4, various machine learning methods that can be used for text classification is explained.

Chapter 5 includes the deep learning models, model structures, data set used, and model parameters.

Chapter 6 includes the experimental setup and the results of the experiments applied on RCV1-v2 data set.

Chapter 7 includes the conclusions of the study.

CHAPTER II

RELATED WORK

The first study on RCV1-v2 data set was made by the authors that created the data set by making corrections on the original RCV1 data set. Commonly used supervised learning techniques studied in text classification such as: SVMs, weighted k-Nearest Neighbour (kNN), and Rocchio style algorithms were used in their study [2]. In [3], the authors propose to use a simple artificial neural network that utilize recent techniques used for NLP problems such as rectified linear units as activation function, dropout to avoid overfitting, and adagrad (adaptive gradient algorithm) as an optimization algorithm. They show that neural networks can reach accuracy levels as well or even better than state of the art. In [4], they use CNNs as a solution for text classification problem in a semi-supervised framework. As an input to the CNN model, they learn embeddings of small phrases instead of word embeddings from unlabeled data. In [5], the authors show that applying deep learning to extreme multi-label text classification (xmc) with CNN models that integrate multi-label co-occurence patterns as additional features can give accurate results. In [6], they convert text to a graph which model text by using words as vertices and the relation between words as edges. They use the graph as an input to the CNN model and use convolution operations on this data. In [7], the authors use a binary classifier for each different category. As a base model, they use gru and integrate attention to their model. They reuse model parameters tuned at higher levels for classification task at lower level labels. In [8], the authors represent a transformer based model. The transformer model used for the study has multi head attention at the word, sentence, and graph levels. They use a graph data structure which can model the words and their relations. They give the graph as an input to the transformer model. The hierarchy of the labels is also considered by using label sequences for one text document. The label sequence is turned into vector space by using skip-gram technique. After getting word embedding for each label, they

integrate these label word embeddings into weighted loss function to get the features of the text with higher accuracy.

Although RCV1-v2 data set is frequently used in published papers, not all studies work on the full classification problem. Some of these studies [9,10] only use subsets of the data set. Some of the studies work on a full data set, but they do not comply with the training-test split of [1], since the original split may be impractical for some tasks (training 23k instance, test 780k instance). As a result, only some of the studies are directly comparable to our research, so only those studies are shown in the comparison table in the experiments section.

CHAPTER III

TEXT CLASSIFICATION

The text classification problem has been studied in many different fields such as database and data mining in the literature and new methods have been introduced to increase its accuracy. The problem of classification is defined as follows. We have a set of data points in the multi dimensional vector space:

$$D=\{X_1,\ldots,X_N\} \tag{3.1}$$

and each data point can be labeled with a class value from k different discrete values by {1…k}. The classification algorithm uses the training data to learn the relation between the input data features and the class labels. The task of the classification algorithm is to assign any sample in the test data set to the correct class value. In some text classification problems, it is sufficient to assign a single class value to each data point in the test data set, while in other classification problems, the probability values of the classes must be estimated; such problems are called multi label classification [11]. In some types of the classification problem, the model ranks the different class values to the data point or assigns multiple class values to the same data point [12] in the test data set.

3.1 SINGLE-LABEL CLASSIFICATION

Single label classification is a method that assigns any input sample to a single label from a sample cluster C that has disjoint labels. The single-label classification can be analyzed under two sub categories: Binary and multi-class classification [13]. If the input data is assigned one of the two class values then it is called binary classification. If data is assigned to one of multiple class values then it is called multi-

class classification. The simplest classification method is binary classification and is the lowest requirement for a method to be called classification. All classification methods basically work by making binary classifications, so it is possible to run any classification algorithm as a binary classifier. In the literature, there are several methods that propose solutions to the multi-class classification problems [14].

3.2 MULTI-LABEL CLASSIFICATION

Since multi-label classification is used in the most important application areas such as classification of news item data set, it has gained a serious importance over time. In multi-label classification, more than one class label value is assigned for each data sample. In contrast to single-label classifications, where a single class is assigned to each data point in the test data set, in the multi label classification method, more than one label is assigned to each data point, for example news item documents with more than one parent and child category. As opposed to single-label classification, each input sample is associated with a set of target labels in multi-label classification. The number of class labels assigned to each data sample varies. For example, in a news item data set classification process, some news items may be in a few sub categories, while some other news items may be in both a few sub categories and a few higher categories [14]. This unpredictable nature of multi-class classification problems cause an increased complexity in computational tasks [15]. For multi-label classification, machine learning methods used in binary classification are used in one versus all mode and modified to assign more than one class value. Artificial neural networks are also used in classification mode for such problems. Several methods have been developed for multi-label classification and is available in the literature. These methods will be presented in chapter 4.

CHAPTER IV

MULTI-LABEL TEXT CLASSIFICATION TECHNIQUES

In this chapter, we discuss the most popular techniques of text classification. First, we start with a traditional method called rocchio and this classification technique is explained. Next, ensemble-based learning techniques such as boosting and bagging, which combines weak classifiers to produce a more accurate classifier than a single weak model would, is explained. Next we cover logistic regression that uses a logistic function to model the conditional probability. Later a brief overview of naïve bayes classifier, that is a probabilistic classifier based on Bayes' theorem that suggests statistical independence, is given. Non-parametric techniques were also used for document classification such as k-nearest neighbor (KNN). Support Vector Machines (SVMs) is one of the supervised learning methods used for classification and it works by drawing a line to separate the data points in the multi-dimensional space. We can also see that classification algorithms such as decision tree and random forest are used extensively in classification problems. Lately, deep learning methods have shown successful results with text classification.

Classification methods like rocchio classifier, logistic regression and SVM are natively equipped to perform binary classification tasks but there are techniques which makes multi-label text classification possible. This is the reason that they are also explained in this chapter.

4.1 ROCCHIO CLASSIFIER

The Rocchio algorithm, which is used to obtain the desired data by querying regularly structured data such as full text databases, was found

by J.J. Rocchio in 1971 [16]. Since then, many researchers have studied and further developed this technique for classifying texts and documents [17,18].

The main idea behind Rocchio algorithm is to maximize the query vector to the documents that have semantically in relation and minimize the query vector to the less semantically related documents. For example, if we want to classify the news items data set with Rocchio classification algorithm, first we need to convert each document to a numerical representation. We can achieve this by calculating tf-idf vector for each news item document. After calculating tf-idf vectors, since we know which vector belongs to which categories, we should calculate a generalized vector for each category by combining the news item vectors that are in the same class. This vector is calculated as follows :

$$\vec{\mu}(c)=\frac{1}{|D_C|}\sum_{D\epsilon D_C}\vec{V}(d) \tag{4.1}$$

where DC is the set of all documents that belong to class c and v(d) is the vector space representation of d.

To classify each news item data, the classification rule is to assign the data point to the nearest centroid in the vector space. This centroid is in a region and each region represents the class value that the model predicts the document to be assigned to.

4.2 BOOSTING AND BAGGING

In almost all of the classification problems, the aim is always to get more accurate results than the previous experiments. However, it may not always be possible to introduce a new classification model. In such a case, a better decision making model can be created by using the decisions made by the previous models. The algorithms that make this concept possible are called ensemble algorithms. Voting classification techniques which are examples of ensemble learning, compared to base classifiers, makes a boosting effect by reducing multiple classifier errors, thus achieving higher accuracy rates in classifying text data [19]. While boosting adaptively changes the training set distribution based on the performance of previous classifiers, bagging does not consider the previous classifier [20].

If we apply Bagging algorithm to a news item data set, the first step will be to get subsamples form the training data set. After we get the subsamples, we start

training weak classifiers on these subsamples. The weak classifier models work in parallel. The final predictions on the validation set is obtained by combining all the predictions that come from the weak classifiers. Now we have a bagging algorithm that can make predictions for the test part of our news item data set.

Boosting method is mostly used with many decision tree classifier ensembles. The main idea for this algorithm is to give a higher weight for the misclassified data samples and reduce the weight of the less frequently misclassified data samples so that a more accurate and more sensitive classifier will be obtained. For instance, let us assume that we will use boosting algorithm on a news item data set. The first step will be to generate subsample from the training part of our news item data set. As a second step we choose a weak classifier and start training the weak classifier on our news item subsample data set. We give higher weights for the misclassified data points in our subsample. Next, we choose another model that tries to make corrections on the previous classification results. We will add many more models to make corrections on the misclassified data points. The final model that is used for training will be the one with the weighted average of all the other models.

The bagging algorithm was introduced by L. Breiman [21] in 1996 as a voting classifier method. The algorithm was created with the idea that the predictions made by more than one classifier on a data point in the same problem would be more accurate if selected with the majority vote from the same base classifiers [20]. Bagging algorithm often considers homogeneous weak learners, learns them independently in parallel and combines them according to some sort of deterministic averaging process.

4.3 LOGISTIC REGRESSION

Considering the text classification problem, logistic regression is a technique that creates the least costly line equation so that each textual document can be classified linearly or non-linearly on the plane. LR was introduced and developed by statistician David Cox in 1958 [22]. Using LR, we can make predictions for numerical values or make classification on categorical values. For instance, we can use LR to make classification on our news item data set. LR can help us label each news item with discrete news category labels. We can also say that logistic regression lays a foundation for deep learning models and tries to find the best S in the multi-

dimensional vector space and this distinguishes it from its close sibling linear regression. We can apply LR algorithm to a news item data set. Let us assume that our LR algorithm tries to find the best weight and bias values. To find the most optimal values, LR gives the sum value to sigmoid function. We prefer sigmoid as an activation function since it produces probabilistic values between zero and one. In our example, if LR produces a number very close to one and this value is higher than our threshold, then we can conclude that this news item belongs to economy class value. The sigmoid function is defined as:

$$\text{sigmoid}(n)=\frac{1}{1+\exp^{-n}} \tag{4.2}$$

We can use Logistic regression not only for binary classification problems but also for multi class and multi label classification problems.

4.4 NAİVE BAYES CLASSIFIER

When we want to assign class labels for a data consisting of text documents, the naïve bayes classifier is a simple but well-working method that assumes the effects of each word on the classification accuracy occur independently of each other. The Naïve Bayes classification method is theoretically based on Bayes theorem, which was formulated by Thomas Bayes between 1701-1761 [23,24]. The naïve bayes algorithm is described as follows:

$$P(A|B)= \frac{P(B|A)P(A)}{P(B)} \tag{4.3}$$

Naïve Bayes algorithm helps us assign class values to text documents by making calculations that follow statistical rules. To understand the practical use of naïve bayes algorithm, let us assume that a data set that has news items is given to multinomial naïve bayes algorithm for classification. The first step will be to calculate the rate of each category in the training data set. After this step, we should calculate each word being in each category in the training data set. In this step, we make our calculations with the assumption that each word is independent from other words in the data set. When we complete the calculation of conditional probability of each word, we can use these probabilities to calculate each document probability for the

assignment to the news item class value. Finally, we can use a threshold value for the probability values to determine the suitable categories for each news item document.

4.5 K-NEAREST NEIGHBOR

The k-nearest Neighbors algorithm (KNN) is a non-parametric technique used for classification. This method has been used for text classification applications in many research areas [25] in the past decades.

The main idea about the KNN algorithm is that this algorithm does not have a training phase and this algorithm works on extracting the input features and representing the document as vector in the multi-dimensional space. After the vector is placed on the multi-dimensional plane, the k nearest classes are examined and the document is assigned to the class value with the majority match. Let us assume that we apply this algorithm practically on data set of news items. In this case, each news item in the training set is placed on the multi-dimensional space. After, each news item in the test data set is put into the plane, each one is compared in terms of a distance algorithm to the data points in the training data set and assigned to the majority class. Here comes the following question about the algorithm: what value should be chosen for k and what distance algorithm should we use? To answer this question we need to apply parameter tuning. The decision rule of KNN is :

$$f(x)=\text{argmax } S(x,c_j) \tag{4.4}$$

where S refers to score value with respect to $S(x,c_j)$, the score value of candidate i to the class of j, and output of $f(x)$ is a label to the test set document. The performance of KNN is depends on finding a reasonable distance function, which makes this technique a very data dependent algorithm [26,27].

4.6 SVM

Support Vector Machine (SVM) was originally developed for binary classification tasks. However, many researchers work on multi-class problems and use this dominate technique [28]. The figure 2 shows the SVM algorithm working in two different modes : linear and non linear separation in a two dimensional space.

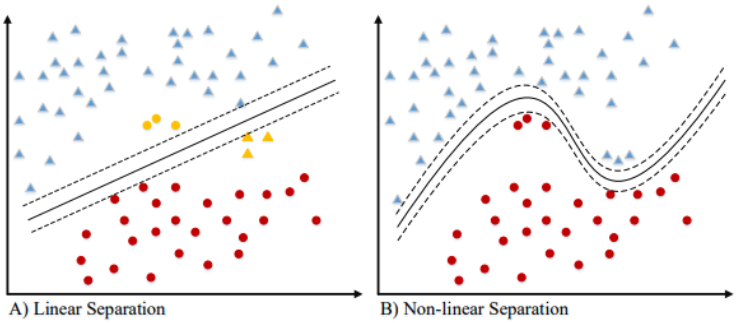

Figure 4.1: This figure shows the SVM algorithm in linear and non-linear mode for 2 dimensional data set.

SVM is an algorithm that is categorized under classification methods but it is also used for regression purposes. The aim of the SVM algorithm is to find the best separating linear or multi dimensional plane to make the most accurate binary classification. If we use the SVM algortihm to classify a data set containing news items, SVM uses the following steps. In the first step, if we give tf-idf vectors of the news documents to the model, SVM will create data points in the multi dimensional space for each document. The dimension of the space will be equal to the number of unique words in all of the documents. SVM will try to find the best separating line to find the distinct class values for the news items. As the space is multi dimensional SVM will use kernel trick and find the best hyper plane that separates the distinct class values. After finding the separating hyper plane, SVM will create data points from the test part of the news items data set and predict the class values according to the separated plane.

4.7 DECISION TREE

One earlier classification algorithm for text and data mining is decision tree [29]. Decision tree can be used for classification (most common use) or regression problems. In decision tree algorithms, there are decision nodes that divide the data according to various parameters. Let us suppose that there is a data set that consists of news items and we want to classify this data set with decision tree algorithm. Decision tree will generate a root node for all the news items from the training data set. The root node will divide the data according to a parameter. If all the data samples passing through the same node belongs to the same class value then this node will end with a leaf node. If the data samples do not belong to the same class value, then a new

parameter will be chosen by the decision tree and more decision nodes will be used. The training phase will continue until all the data points from the training set will end in the same leaf node.

Sometimes branches that use less important features may need to be removed from the model. As a result of this process, the complexity of the model decreases and its performance on the validation data set increases.

4.8 RANDOM FOREST

Random forest algorithm consists of decision trees that are trained from subsets of the training data set. These decision trees are randomly selected by the random forest algorithm. Random forest decisions are created by taking the average of decision trees. Random forest algorithm can also reduce the overfitting problem of decision trees by taking the average of the results. If we apply this algorithm on a news item data set, the first step will be to select random samples from the training data set. After, Random forest will create a decision tree for each of the sample. In the third step, Random forest will take a prediction from each of the decision trees. For classification problem random forest will use bagging method and will select the most voted prediction. After the training phase, random forest will use the trained parameters for the test data set predictions. This method, which used t tree as parallel, was introduced by T. Kam Ho in 1995 [30]. The general structure is shown in the figure below :

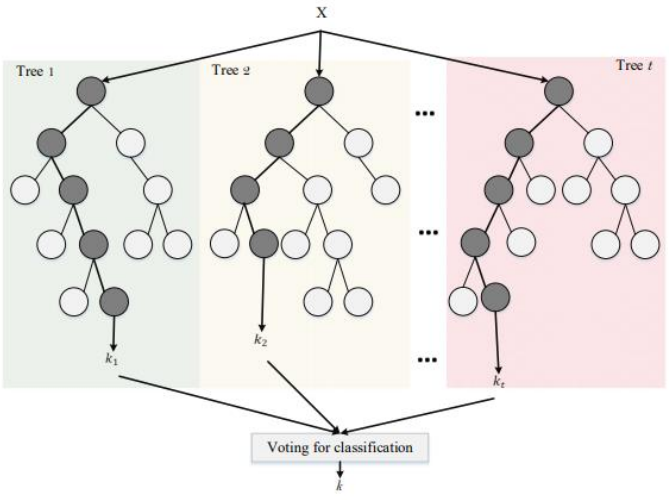

Figure 4.2: Random Forest.

4.9 DEEP LEARNING

Deep learning is a technique that does not need the processing of the input data and trains itself to predict the correct outputs by learning the weights in the middle layers. We can say that deep learning is comparable to human brain in terms of the way that it sends and changes input data through neurons with applying activation functions. When we compare deep learning models with other machine learning methods, it is seen that these models have achieved significant success in many areas including text classification due to the increase in the amount of data that is produced daily.

If we summarize the steps that take place when a deep learning algorithm is run on a data set that consists of news items as an example, in the first step we should convert each news item document into word2vec embedding vectors. After embedding vectors are obtained, each dimension of the vector is sent to input neurons of the deep learning model. We start the forward propagation step to find the most optimal weight values that will minimize the loss function. Our loss function is calculated by comparing the predicted class values with the ground truth values. These values are news category names like economy, sport, or subcategories such as CCAT. To minimize the loss function deep learning model will back propagate by finding the derivatives and using learning rate to tune the leaning speed of the algorithm. In the final output layer we can use sigmoid function to find probabilities between zero and one and applying a threshold values we can pick the category labels to carry out the multi label classification process.

In this study we use RCV1-v2 data set which contains more than 800000 news items and it must be noted that such a data set when working with a deep learning model requires high main memory and graphics processing unit memory for data processing. This high resource requirements is a disadvantage for deep learning models.

When used in areas covered by natural language processing, deep learning algorithm is a method that can give at least as good accuracy as other algorithms without the need for pre-processing of data before classification or feature extraction techniques such as part of speech tagging [31], especially in topic classification, sentiment analysis, question answering [32] and language translation [33,34].

CHAPTER V

METHODOLOGY

In this section, we first describe the different deep learning models that are used to classify the RCV1-v2 data set starting from models that were used for a long time to classify textual data like flat deep learning models, cnn, rnn, lstm to recently developed models like attention and transformer models. Secondly, a highly imbalanced data set which is called RCV1-v2 is described. Finally, the model parameters and the evaluation metrics are specified.

5.1 DEEP NEURAL NETWORKS

Deep neural networks (DNN) are neural networks that have at least 2 or more hidden layers. A DNN has one input layer, two or more hidden layers and an output layer. Every hidden layer in a DNN receives input from previous layer, multiplies the data with a weight, applies an activation function and has a connection to the next layer. The weight of the hidden layer node shows the importance of the input to the output. Two steps take place in the work of a DNN. The first step is called forward propagation. In this step, the DNN model gives output or outputs for classification or regression problems. The second step is back propagation. In the back propagation step, the derivatives for intrinsic parameters of the model are calculated according to the loss function. [11]. Figure 5.1 shows the general structure of a deep neural network.

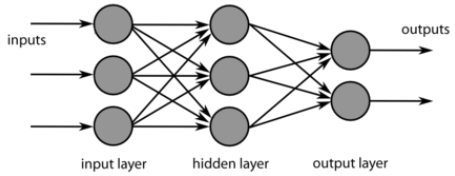

Figure 5.1: A sample deep neural network.

The weights of the nodes that are in hidden layers of the DNN is tuned in the back-propagation step. The input layer consists of features that are determined for the problem. Bag of words, TF-IDF or word embedding vectors are examples of feature sets for NLP problems [12]. In the output layer, the number of nodes must be equal to the number of classes to represent each class value for multi label or multi class classifications. For binary classifications, the output layer can only have a single node. For hidden layers sigmoid (1) or RELU (2) is commonly used as activation functions [12].

$$f(x)=\frac{1}{1+e^{-x}} \in (0,1) \quad (5.1)$$

$$f(x)=\max(0,x) \quad (5.2)$$

5.2 CONVOLUTIONAL NEURAL NETWORKS

A convolutional neural network is a deep neural architecture that is commonly used for hierarchical classification of documents [13,14]. CNNs were created with the intent of image processing however there are many successful examples of the use of CNNs in text classification [1,15].

In a basic CNN has two different layers. The first layer is called convolutional layer. In this layer, a kernel of size d x d is used on the pixels of the image with a stride value. The importance of the convolution operation is that at each operation different features are captured from the image. While at lower levels edges or colors are captured, at higher levels shapes are captured from the image. The kernel depth is always same with the depth of the image. If the image has red, green and blue channels, then the kernel will be applied for each channel. The second layer is called pooling layer. The pooling layer reduces the spatial size of the image tensor. This operation helps to reduce the computational power that is required for the CNN model. There are two types of pooling that is commonly used in the literature. Max pooling and average pooling. Max pooling returns the maximum value that is covered by the pooling kernel from the image. Average pooling returns the average of the values that is covered by the pooling kernel from the image. Finally the image tensor is flattened with high and low level features to make a final prediction for the problem.

5.3 RECURRENT NEURAL NETWORKS

Recurrent neural network (RNN) is another architecture that has been used by researchers for text mining and classification problems [16,17]. RNN works on sequential data. Text can be considered as a sequential data when we consider each word as an input to a layer and its output will have an influence both in the next layers and in the final output. RNN shares the same parameters for each time step and this causes for the model to take sum of the errors in the back–propagation step. As a result, this technique is considered as a method that gives accurate results when used for text, string, and sequential data such as time series. Researchers use RNN mostly with LSTM or GRU layers. The problems of vanishing and exploding gradient occur in the context of RNNs since the derivatives are multiplied through the layers and this may cause the gradient approach to zero or increase exponentially when the model is back-propagated.

5.4 LONG SHORT TERM MEMORY

Long short term memory (LSTM) is considered as a variant of RNN. While RNNs are good at making classifications with recent information, they are weak at storing long term memory. LSTM has cells that store additional information and it also has gated cells that manipulate these data cells. LSTM is considered to be useful for avoiding the vanishing gradient problem that occurs in many text classification problems when the sentence structure is too long in terms of word count [18].

To prevent vanishing and exploding gradient problem, LSTM uses memory cells. These memory cells stores previous data and they also decide which data will be preserved and which data will be deleted. Using this approach, we can protect long dependencies in data series.

LSTM also has gates that control the flow of data in or out of the cells. The input gate controls whether the data will move into the cell or not, the forget gate helps to decide which information to store or throw away and the output gate makes decisions on the previous hidden states. The gates in the LSTM model use sigmoid activation function since we want positive values to determine the importance of the feature. “0” means the gates do not allow the flow of data and “1” means the gates allow the flow of the data completely. The mathematical equations for the input, forget

and output gates are as follows:

$$i_t = \sigma(w_i[h_{t-1},x_t]+b_i) \quad (5.3)$$

$$f_t=\sigma(w_f[h_{t-1},x_t]+b_i) \quad (5.4)$$

$$o_t = \sigma\,(w_o[h_{t-1},x_t]+b_o) \quad (5.5)$$

5.5 BI-LSTM

We can say that Bi-LSTM actually has two independent rnns that work together. Using Bi-LSTM, it is possible to have forward and backward information at each time step. Bi-LSTM generally show better results when compared to LSTM as they can understand the context better.

5.6 GATED RECURRENT UNIT

Gated recurrent network (GRU) is another variation of RNN. GRU is very similar to LSTM except that GRU does not have the cell state and it only has two gates; update gate and reset gate.

5.7 BI-GRU

Bi-GRU consists of two GRUs. One of the GRU takes the input in a forward direction and the other GRU takes the input in the reverse order. We can consider bi-GRU as a normal RNN that has two forget and reset gates.

5.8 ATTENTION

For encoding sequence data, we used the Attention model. It is a simple model which is used in encoder-decoder models. Instead of producing context vector from the entire encoding sequence data, the attention model calculates the context vector by using the weighted sum of the encoding sequence data. The weight is calculated from a model that scores how similar the input and the output match. It has been very successful in domains not only in encoder-decoder models but also in text classification. The attention model was first introduced into neural machine translation [20]. Some variations of the attention model is used in the literature. One of these variations is hierarchical attention networks.

Let's consider each document as being composed of characters, words, and sentences. A hierarchical attention model can be created either bottom-up (word to sentence) or top-down (sentence to word) to extract data on different levels. Bottom-up construction is used in document classification [21].

5.9 TRANSFORMERS

Transformers was first introduced by the "Attention is all you need" paper, implying that the transformers model is a model that uses self-attention and multi head attention methods to find the relationship of the word given as input in both directions with other words, without the need for models such as rnn which uses sequence aligned data and cnn which performs convolution operations on middle layers [22]. There are three possible kinds of attention in a model. Encoder-decoder attention is the attention between the input sequence and the output sequence. Self-attention creates connections from different positions in the input sequence to make a representation from the same sequence. It is also called as intra-attention.

There are language representation models (BERT, Roberta, XLnet) that are designed to pre-train deep bi-directional representations which is different than other methods since it captures the context both from left and right. These pre-trained models can be used in other domains like text classification by adding a layer on top of the core model as a CLS token and fine-tune the model in next sentence prediction mode [23].

5.10 DATA SET

We use RCV1-v2 data set in our experiments. There are more than 800,000 manually labeled news stories in RCV1 data set. These news stories have a hierarchical structure in the assigned labels that has six levels. A researcher can work on the assigned topic codes, industry codes or region codes to make text classification. There are corrections to the original data, which is referred to as RCV1-v1, and the corrected version is called RCV1-v2. There are a total of 804,414 RCV1-v2 documents, divided into 23,149 training documents and 781,265 test documents. This division is called LYRL2004 split [2]. We use this split in our experiments. You can see the data set statistics in the Table 1.

Table 5.1: An overview of statistical data on the RCV1-v2 data set: P is the number of training instances, Q is the number of test instances, R is the total number of features, S is the total number of class labels, S' is the average number of labels per document, ~S is the average number of documents per label, T' is the average number of words per document in the test set.

	RCV1-v2 Data Set Statistics							
Data Set	P	Q	R	S	S'	~S	T'	~T
RCV1	23,149	781,265	47,236	103	3,18	729,67	259,47	269,23

5.11 MODEL

5.11.1 Model Parameters

We apply early stopping as a callback function to avoid the model from overfitting. This helps to get the best internal state of the hidden layer parameters and other statistics during training. Another callback function applied for each model is reducing the learning rate when the model does not improve over a certain period, like 3 or 5 epochs. We think this is a good practice since a minimum can only be reached when the learning rate approaches zero. Otherwise, with a permanent large learning rate, the model will bounce around the minimum and always overshoot it.

When deep learning models are run on the training set, the input data is sent to the model in batches of 16, 32 or 64 for the optimization algorithms to work more effectively. When a batch run is made in this way, the number of words for each document entering the model should be equal. In this study, the value of 300 (max sequence length) was used as the maximum number of words that is sent to deep learning models. If the sequence length is less than 300, then 0 is added to the end of the sequence for each missing dimension and if it is more, it is truncated.

It can also be inferred from the literature that it is recommended to use as low learning rates as possible for pre-trained transformer models. In this study, 2e-5 learning rate was used in line with this recommendation. In addition, weight decay parameter with a value of 0.01 is used to reduce the learning rate as it approaches the optimization target for each experimental study when using pre-trained transformer models.

5.11.2 Activation Functions

We prefer to use ReLU as an activation function in the internal hidden layers. ReLU has two advantages when compared to other activation functions. The first one is the reduced likelihood of the gradient vanishing, which occurs when the output of the internal node of the hidden layer is greater than zero. In this situation, the gradient will always be constant. The second one is sparsity. When the output of the internal node is less than or equal to zero, ReLU will output a more sparse representation of the data. Sparse representation is always preferred to dense representations. In the output layer, sigmoid activation function is applied for each output node. Sigmoid always returns an output between zero and one. Giving a threshold value, we can decide which class labels to assign to documents.

5.11.3 Optimization Functions

We use adam optimization function for every model that we apply on RCV1-v2 data set. Adam is summarized by [24] as an algorithm that needs little memory to work, is not effected by the rescaling of the gradients and gives highly accurate results on problems that are large in data or parameters. Adam optimization algorithm has some benefits when compared to other algorithms. First, adam can compute adaptive learning rates for different parameters. However, stochastic gradient descent (SGD) maintains only one learning rate through the training process. Second, adam optimizer combines the best properties of the AdaGrad and RMSProp at the same time, therefore Adam works well on sparse gradients when the porblem data is noisy. Third, adam optimization works well in noisy objectives with high dimensional parameter spaces.

5.11.4 Frameworks

All the models except the pre-trained transformer models were implemented in keras api with tensorflow backend. Transformer models were implemented in pytorch hugginsface library.

5.12 EVALUATION METRICS

After the deep learning model for the problem is selected and the experiment is completed, finally, the evaluation of the results should be started. Comparing evaluation results in text classification experiments is a challenge since separate

experiments use different evaluation metrics. We prefer to start with the accuracy metric as it is the simplest and one of the most common methods. In addition to this, we use micro and macro averaged F1. In some experiments [5,25-27], ranked-based metrics like precision@k and normalized cumulative discounted gain@k are also used. Therefore we include p@k scores as well.

5.12.1 Accuracy

To better understand the classification accuracy, we can reduce the problem to a binary classification problem. Let us assume that in our problem we only have one topic category namely ECAT (economics). Each news item in RCV1-v2 either belongs to ECAT or does not belong to ECAT. Then, the classification accuracy will be the ratio between the number of correct predictions and the total number of input samples. The number of correct predictions includes the news items that are in ECAT category and classified as ECAT and the ones that are non-ECAT and classified as non-ECAT. This measure works well only if there are equal number of samples belonging to each class. The mathematical formula for this evaluation is described in (3).

$$\text{Accuracy}= \frac{TP+TN}{TP+TN+FP+FN} \tag{5.6}$$

True Positives and True Negatives are the elements that are on the main diagonal of the confusion matrix and there is agreement between the labels assigned by the algortihm and the data set ground truth labels. The denominator of the equation has all the elements from the confusion matrix. All correctly classified and incorrectly classified elements are included in the denominator.

5.12.2 Micro Averaged F1

In almost all classification studies, we want the measurements obtained as a result of the experimental study to be high, including precision and recall. However, there is a tradeoff between precision and recall. While working on a classifier, improving the precision score mostly lowers the recall score and vice versa. If we would use a single metric to compare two different classifiers, then we can use F_1 score. Micro averaged F_1 score measures the contribution of all labels in the data set.

When we consider this property of micro averaged F_1 score, it may be inferred that common root labels like ECAT will have more importance from this measure. Even if the classification algorithm has a low accuracy on the rare labels, the micro F_1 score will still be high. The mathematical formula for F_1 score is shown in (5.7), precision in (5.8), and recall in (5.9).

$$F_1 = 2 * \frac{\text{Precision*Recall}}{\text{Precision+Recall}} \tag{5.7}$$

$$\text{Precision} = \frac{TP}{TP+FP} \tag{5.8}$$

$$\text{Recall} = \frac{TP}{TP+FN} \tag{5.9}$$

If we consider multi-class classification, then F_1 score should include all the classes. To calculate this, we need a multi-class measure of precision and recall to be inserted into the harmonic mean. There are two different forms in the literature giving rise to macro-F_1 and micro F_1. The mathematical formula for micro-F_1 is shown in (5.10), micro-precision in (5.11), and micro-recall in (5.12). In equation 5.11, total column is considered as precision and in equation 5.12, total row is considered as recall.

$$\text{Micro } F_1 = 2 * \frac{\sum_{k=1}^{K} TP_K}{\text{Grand Total}} \tag{5.10}$$

$$\text{Micro Average Precision} = \frac{\sum_{k=1}^{K} TP_K}{\sum_{k=1}^{K} \text{Total column}_K} \tag{5.11}$$

$$\text{Micro Average Recall} = \frac{\sum_{k=1}^{K} TP_K}{\sum_{k=1}^{K} \text{Total row}_K} \tag{5.12}$$

5.12.3 Macro Averaged F1

To get the macro averaged F_1 score, first, we need to compute macro precision and macro recall. They are computed by taking the average precision for each predicted class and average recall for each actual class. As a result the macro-F_1 will give similar importance to each class label. Some labels are rarely assigned to news items and some labels are almost assigned to the majority of the news items. Therefore we can say that macro-F_1 score will be low with the classifiers that perform well on

common labels such as root categories like ECAT while performing weakly on rare labels such as leaf nodes like GWELF (welfare, social services).As a result, the macro F_1 approach considers all the classes as basic elements of the calculation. The mathematical formula for macro F_1 is shown in (5.13), macro precision in (5.14), and macro recall in (5.15). In equation 5.13, subscript M denotes macro averaged version of precision and recall metrics and in equations 5.14 and 5.15, k denotes a generic class and the mathematical formulas are applied from k=1 to K for single classes.

$$\text{Macro } F_1 = 2 * \frac{\text{Precision}_m * \text{Recall}_m}{\text{Precision}_m * \text{Recall}_m} \tag{5.13}$$

$$\text{Macro Average Precision} = \frac{\sum_{k=1}^{K} \text{Precision}_k}{K} \tag{5.14}$$

$$\text{Macro Average Recall} = 2 * \frac{\sum_{k=1}^{K} \text{Recall}_k}{K} \tag{5.15}$$

5.12.4 Precision at K

Precision at k is a rank-based evaluation metric to present the quality of a short ranked list of relevant labels for each test instance. In this study, we calculate precision at k for k =1,3,5. If we denote, $y \in \{0,1\}L$ as the vector of true labels of a document, and $y' \in RL$ as the score vector predicted by the system for the same document, the mathematical formula is (5.16).

$$P@k = \frac{1}{k} \sum_{l \in r_{k(y')}} yl \tag{5.16}$$

where rk(y') is the set of rank indices of the really relevant labels under the top-k part of the ranking predicted by the system for a document. We can interpret the precision@k score like this; suppose precision at k=3 for the topics classification task is 90%, this means that 90% of the news items have all the relevant labels from the top 3 most frequent labels list from the ground truth labels.

CHAPTER VI

EXPERIMENTS

6.1 EXPERIMENTAL SETUP

For our experiments, we worked on jupyter notebook and used the google-cloud colab platform that supports python 3 and comes with pre-installed deep learning frameworks such as keras api with tensorflow backend and pytorch. The notebook that is provided by colab supports hardware accelerators like gpu and tpu that have optimizations on tensor operations. We configured the notebook to use gpu as the hardware accelerator and implemented the code in the python 3 execution environment, as shown in Table 6.1

Table 6.1: System Specification Setup in COLAB

SPECIFICATIONS	
GPU	1 x Tesla K80
CPU	1 x single core hyper threaded Xeon Processor @2.3
RAM	25 gb Available
DISK	33 gb Available

During the experiments, we used different models with mini-batch optimization steps. When using mini-batch, we had to keep sequence length to a constant value. However, sequence length that is higher than a threshold gives memory exhausted errors in colab environment. Another challenge we faced was to ensemble transformer models. When we tried to ensemble transformer models, we faced with gpu memory exhausted errors.

The jupyter notebook files that contain the source code for these studies can be found in Github repository$_1$[1].

[1] https://github.com/scorpio11/Multi-label-Classification

6.2 RESULTS

We evaluated the models with respect to flat accuracy, micro-F1, macro-F1, and p@K metrics. We also show the related work scores in similar metrics in Table 6.2 to compare the baseline to the state of the art.

Table 6.2: Experimental Results from Literature

RCV1-v2	Mi-F1	Ma-F1	Acc	p@1
Lewis et al. (2004) SVM1	.816	.607	-	-
Lewis et al. (2004) SVM2	.810	.546	-	-
Lewis et al. (2004) kNN	.765	.549	-	-
Lewis et al. (2004) Rocchio	.693	.495	-	-
Nam et al. (2015) $NN_{(A)}$	.8385	.6457	-	-
Nam et al. (2015) $NN_{(AD)}$	.8397	.6404	-	-
Johnson & Zang (2015) CNN	**.857**	.671	-	-
Liu et al. (2017) XML-CNN	-	-	-	**.9686**
Peng et al. (2018) HR-DGCNN-3	.7618	.4334	-	-
Banerjee et al. (2019) GRU	.7654	.4842	-	-
Gong et al. (2020) F-TRANSFORMER	-	-	-	.8933
Gong et al. (2020) F-TRANSFORMER-W	-	-	-	.915
Gong et al. (2020) HG-TRANSFORMER	-	-	-	**.958**

As shown in Table 3, we can consider Lewis et al. benchmark results on RCV1-v2 data set as baseline classifiers. Although it is difficult to compare studies with each other as they use different evaluation metrics, it could be inferred from Table 3 that Johnson & Zang et al. (2015) has the highest micro-F1 score and Liu et. al (2017) and Gong et. al. (2020) have highest p@k=1 scores respectively.

In our study, we start text classification with simple CNN and RNN models, continue with applying attention on words and sentences and finally fine tune the pre-trained transformer models. Table 4. presents the results of our study.

Table 6.3: Experimental Results from the Study

RCV1-v2	Mi-F1	Ma-F1	Acc	P@1
CNN (Glove)	.77	.41	.43	.85
CNN (Fasttext)	.77	.38	.45	.86
RNN (Glove)	.58	.16	.24	.74
RNN (Fasttext)	.57	.14	.23	.76
LSTM (Glove)	.78	.37	.50	.87
LSTM (Fasttext)	.76	.33	.46	.86
GRU (Glove)	.79	.38	.49	.90
GRU (Fasttext)	.77	.34	.46	.87
BI-LSTM (Glove)	.79	.39	.49	.87
BI-LSTM (Fasttext)	.76	.32	.45	.85
BI-GRU (Glove)	.81	.45	.52	.89
BI-GRU (Fasttext)	.78	.37	.47	.88
LSTM-ATTENTION (Glove)	.80	.42	.50	.85
LSTM-ATTENTION (Fasttext)	.77	.33	.45	.86
GRU-ATTENTION (Glove)	.81	.44	.52	.88
HAN (Glove)	.81	.48	.53	.88
HAN (Fasttext)	.77	.36	.46	.86
BERT-TRANSFORMER	.8637	.54	-	**.9148**
ROBERTA-TRANSFORMER	.8633	.56	-	.9065
XLNet-TRANSFORMER	**.8693**	.60	-	.9140

RCV1-v2 topics data set is an imbalanced data set. Some news item categories have a high frequency in the data set. This implies the importance of micro and macro averaged F1 scores. High-frequency categories dominate Micro averaged measures. We think that classifiers with a higher micro-F1 score, in general, will be more successful at classifying at a broader level with more distinct labels. Macro averaging gives equal weight to each category and therefore is affected by low-frequency categories. A text classifier algorithm with a high macro averaged F1 score will be more successful at identifying leaf node categories.

We observe from Table 6.3, that only glove vector representations show higher scores compared to fastText. Also, the highest micro-F1 score is almost achieved in every RNN model with Glove embedding compared to fastText.

In general, transformer models outperform all the other methods in text classification task. This is due to the reasons that transformer models use attention for bi-directional training, and in the fine-tuning approach, we add a dense layer on top of the last layer of the pre-trained models and then train the model on the specific data set.

CHAPTER VII

CONCLUSION AND FUTURE WORK

In this study, we explore the effect of using state-of-the-art deep learning methodologies on multi-label text classification. We implement different neural network architectures such as convolutional neural networks, recurrent neural networks, long short term memory, gated recurrent unit, attention, hierarchical attention networks and fine tune the pre-trained transformer models. We trained the Glove and fastText pre-trained vectors on the RCV1-v2 data set. Finally, we compared our experiments to the literature. We observe that fine tuning transformer models on the training part of the data set give comparable results to state of the art.

The success of transformer models do not seem surprising at all since transformer model uses self-attention to give weights to input words to understand the relation between words and self-attention method uses completely different mathematical formula from classical attention methods used by rnn models. Second reason is that transformer model uses multi-head attention method so that many different self-attention methods are executed in parallel and their results are concatenated. Third reason is that transformer model does not only use word embedding but also use positional encodings to store position of each word and all that data passes to next layer by residual connections to prevent over fitting.

When we examine table 6.3, it is inferred that in almost all of the different classification deep learning models glove word embedding method works with a higher accuracy rate than fasttext word embedding method. In the experimental study that was carried out here, glove and fasttext embedding methods use different pre trained models that was collected from different data sources. Therefore it seems difficult to reach a definite decision about the performance of the two algorithms for the classification of RCV1-v2 data set. It is worth noting that the glove algorithm

generally maintains the frequency of the use of the word in addition to the embedding vectors and the fasttext algorithm increases performance in generalizability by maintain the n grams of characters by going down to the lower levels of the word.

This study can be extended to cover hierarchical data sets. For example, if a category in the leaf node is assigned to a document by the algorithm, then all the categories in the upper hierarchy level must be assigned to the same document. This additional correction may increase the accuracy of the classifier on the data set.

As a future work to improve the classification performance of the algorithm on the data set, we can use bagging or voting ensemble transformer models and make predictions on the test set. It is important to mention that ensembling two models also doubles the memory requirements. Another approach that can be considered, is to use a deep learning hyper parameter tuning framework to find the most optimal parameters for the transformer models like learning rate, batch size and weight decay. It is also important to note that any work that deals with hyper parameter tuning needs costly computational operations on the data set for trial and error iterations.

REFERENCES

[1] LeCun, Y., Bengio, Y., & Hinton, G. (2015). Deep learning. *nature*, 521(7553), 436-444.

[2] Lewis, D. D., Yang, Y., Russell-Rose, T., & Li, F. (2004). Rcv1: A new benchmark collection for text categorization research. *Journal of machine learning research*, 5(Apr), 361-397.

[3] Nam, J., Kim, J., Mencía, E. L., Gurevych, I., & Fürnkranz, J. (2014, September). Large-scale multi-label text classification—revisiting neural networks. In *Joint european conference on machine learning and knowledge discovery in databases* (pp. 437-452). Springer, Berlin, Heidelberg.

[4] Johnson, R., & Zhang, T. (2015). Semi-supervised convolutional neural networks for text categorization via region embedding. *Advances in neural information processing systems*, 28, 919.

[5] Liu, J., Chang, W. C., Wu, Y., & Yang, Y. (2017, August). Deep learning for extreme multi-label text classification. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 115-124).

[6] Peng, H., Li, J., He, Y., Liu, Y., Bao, M., Wang, L., ... & Yang, Q. (2018, April). Large-scale hierarchical text classification with recursively regularized deep graph-cnn. In *Proceedings of the 2018 world wide web conference* (pp. 1063-1072).

[7] Banerjee, S., Akkaya, C., Perez-Sorrosal, F., & Tsioutsiouliklis, K. (2019, July). Hierarchical transfer learning for multi-label text classification. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics* (pp. 6295-6300).

[8] Gong, J., Teng, Z., Teng, Q., Zhang, H., Du, L., Chen, S., ... & Ma, H. (2020). Hierarchical graph transformer-based deep learning model for large-scale multi-label text classification. *IEEE Access*, *8*, 30885-30896.

[9] Daniely, A., Lazic, N., Singer, Y., & Talwar, K. (2017). Short and deep: Sketching and neural networks.

[10] Duchi, J., Hazan, E., & Singer, Y. (2011). Adaptive subgradient methods for online learning and stochastic optimization. *Journal of machine learning research, 12*(7).

[11] Aggarwal, C. C., & Zhai, C. (2012). A survey of text classification algorithms. In *Mining text data* (pp. 163-222). Springer, Boston, MA.

[12] Gopal, S., & Yang, Y. (2010, July). Multilabel classification with meta-level features. In *Proceedings of the 33rd international ACM SIGIR conference on Research and development in information retrieval* (pp. 315-322).

[13] Tsoumakas, G., & Katakis, I. (2006). Multi-label classification: An overview, Dept. of Informatics. *Aristotle University of Thessaloniki, Greece*.

[14] Er, M. J., Venkatesan, R., & Wang, N. (2016, October). An online universal classifier for binary, multi-class and multi-label classification. In *2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC)* (pp. 003701-003706). IEEE.

[15] Zhang, M. L., & Zhou, Z. H. (2007). ML-KNN: A lazy learning approach to multi-label learning. *Pattern recognition, 40*(7), 2038-2048.

[16] Rocchio, J. (1971). Relevance feedback in information retrieval. *The Smart retrieval system-experiments in automatic document processing*, 313-323.

[17] Partalas, I., Kosmopoulos, A., Baskiotis, N., Artieres, T., Paliouras, G., Gaussier, E., ... & Galinari, P. (2015). Lshtc: A benchmark for large-scale text classification. *arXiv preprint arXiv:1503.08581*.

[18] Sowmya, B. J., & Srinivasa, K. G. (2016, October). Large scale multi-label text classification of a hierarchical dataset using Rocchio algorithm. In *2016 International Conference on Computation System and Information Technology for Sustainable Solutions (CSITSS)* (pp. 291-296). IEEE.

[19] Farzi, R., & Bolandi, V. (2016). Estimation of organic facies using ensemble methods in comparison with conventional intelligent approaches: A case study of the South Pars Gas Field, Persian Gulf, Iran. *Modeling Earth Systems and Environment, 2*(2), 1-13.

[20] Bauer, E., & Kohavi, R. (1999). An empirical comparison of voting classification algorithms: Bagging, boosting, and variants. *Machine learning*, *36*(1), 105-139.

[21] Breiman, L. (1996). Bagging predictors. *Machine learning*, *24*(2), 123-140.

[22] Cox, D. R., & Snell, E. J. (1989). *Analysis of binary data* (Vol. 32). CRC press.

[23] Pearson, E. S. (1925). Bayes' theorem, examined in the light of experimental sampling. *Biometrika*, 388-442.

[24] Hill, B. M. (1968). Posterior distribution of percentiles: Bayes' theorem for sampling from a population. *Journal of the American Statistical Association*, *63*(322), 677-691.

[25] Jiang, S., Pang, G., Wu, M., & Kuang, L. (2012). An improved K-nearest-neighbor algorithm for text categorization. *Expert Systems with Applications*, *39*(1), 1503-1509.

[26] Sahgal, D., & Parida, M. (2014). Object Recognition Using Gabor Wavelet Features with Various Classification Techniques. In *Proceedings of the Third International Conference on Soft Computing for Problem Solving* (pp. 793-804). Springer, New Delhi.

[27] Sanjay, G. P., Nagori, V., Sanjay, G. P., & Nagori, V. (2018). Comparing Existing Methods for Predicting the Detection of Possibilities of Blood Cancer by Analyzing Health Data. *Int. J. Innov. Res. Sci. Technol*, *4*, 10-14.

[28] Bo, G., & Xianwu, H. (2006). SVM multi-class classification. *Journal of Data Acquisition & Processing*, *21*(3), 334-339.

[29] Morgan, J. N., & Sonquist, J. A. (1963). Problems in the analysis of survey data, and a proposal. *Journal of the American statistical association*, *58*(302), 415-434.

[30] Ho, T. K. (1995, August). Random decision forests. In *Proceedings of 3rd international conference on document analysis and recognition* (Vol. 1, pp. 278-282). IEEE.

[31] Collobert, R., Weston, J., Bottou, L., Karlen, M., Kavukcuoglu, K., & Kuksa, P. (2011). Natural language processing (almost) from scratch. *Journal of machine learning research*, *12*(ARTICLE), 2493-2537.

[32] Bordes, A., Chopra, S., & Weston, J. (2014). Question answering with subgraph embeddings. *arXiv preprint arXiv:1406.3676*.

[33] Jean, S., Cho, K., Memisevic, R. & Bengio, Y. On using very large target vocabulary for neural machine translation. In Proc. ACL-IJCNLP http://arxiv.org/abs/1412.2007 (2015).

[34] Sutskever, I., Vinyals, O., & Le, Q. V. (2014). Sequence to sequence learning with neural networks. *arXiv preprint arXiv:1409.3215*.