University of Essex
Department of Mathematical Sciences

MA981: DISSERTATION

A Title based Extreme Multi-label Multi-class Text Classification

Md Mehedi Hasan Raj

Supervisor: Dr Mario Gutierrez-Roig

November 25, 2022

Colchester

Contents

List of Figures

List of Tables

CHAPTER 1

Abstract

Semantic annotation or tagging is a technique for attaching the labels with most relevant documents or articles. We find annotations in Wikipedia or online library systems. In the documents, the title and full text are available. So far, The performance of annotation for the only title is poor compared to the full text. But, it may be competitive, as we show in our experiment. In our experiment, we propose 4 different deep-learning classifiers that give a competitive result. CNN gives the highest f1 score which is 0.492 for the title, whereas full text gives only 0.387. GRU gives 0.488, LSTM gives 0.482, and MLP gives 0.475. All the classifiers give better results than full text. Also, our CNN, LSTM model gives a better result than the previous study result on the title. We also implement a new classifier, GRU, that also gives the competitive result.

CHAPTER 2

Introduction

Extreme Multi-label text classification annotated the labels with the most relevant of the documents. The world is growing fast in technology, and because of that, data is increasing rapidly. Last few decades, data has increased more than a thousand times. It is easy to handle and categorize a small amount of data, but now it becomes more difficult to organize each document manually by humans. We all know about Wikipedia. In Wikipedia, there are more than one billion categories available. One article can be more than one category or label. These are created by the expert annotator. But, nowadays it becomes difficult to do this because of the billion documents. To overcome this problem, Multi-label text classification comes. Multi-label text classification is not like as simple classification. Most of the classifications can be either binary or multi-class. But, the extreme multi-label text classification problem is comparatively more difficult than others because of its huge labels. Not only the problem of a huge number of labels, but some labels have very few training examples. Though it is a very difficult task, there has been significant research on it, and going on. Several machine-leaning approaches have been proposed for extreme multi-label text classification. So far, all proposals can be either traditional machine learning or deep learning. Deep learning has shown great performance in this field.

CHAPTER 3

Related Work

In the paper [7], The authors proposed a modern method for automated semantic document annotation. Unlike, other proposed methods, they do not take the whole text for training, instead, they take only the text's title. Usually, the deep learning model takes too much time in the training, and it becomes more if we use more data. Taking the whole text is always much more expensive. To overcome this complexity, they use only titles. They also show that titles can be a great impact on text classification like the whole text. In their experiment, first, they demonstrate the model using the whole text, then they only use the title to compare the model. The author's works are mainly divided into three main parts. First for feature extracting, then training the model, and finally comparing the result with the full text. For feature extraction or vector representation, they use the Term Frequency and Inverse Term Frequency or TF-IDF method. TF-IDF is a very popular vectorization method for feature representation. They use the different types of traditional machine-learning methods like Bayes, Logistics regression, support vector machine, and multi-layer perceptron(MLP) that is based on deep learning method. For comparing the result, the author uses the popular metric F1 score for extreme multi-label text classification.

In the paper [12], Deep learning is very convenient in text classification and performs well that shows. The author of the paper [7] demonstrates and updates their work using various deep learning methods like CNN, RNN, and ANN. For ANN they use the MLP with 2k units, for CNN they use the KIM-CNN architecture, and for RNN they use the LSTM. Their experiments show that MLP performs better in the Econbiz dataset, and LSTM performs

better in the PubMed dataset. They use a new threshold of 0.20 instead of 0.50 value for converting the probability output value.

In the paper [9], the Hierarchical Label Set Expansion(HLSE) method is proposed for extreme multilabel text classification. This method is divided into three main categories. First is text encoding. For text encoding, they use CNN's Embedding module. The second is feature extraction. A fully connected neural network is used for this. After the HLSE is used for regularising the labels. The third is for input data representation. Multiple word embeddings are used for this. In the first CNN, a kernel size of 5 is set and a 200 filter is used. ReLu is for the activation function and then max pooling is used for reducing the dimensionality. The kernel size is 3 and the filter sizer is 100 for the second CNN with the ReLu activation function. This output is linked to a dense layer with 1024, but before max-pooling is applied. The final output is the number of graphs. The authors use the PubMed dataset for the experiment and got a micro f1 score is 0.55.

In the paper [5], a new method BIGRU with an attention model is proposed for extreme multi-label text classification. This is an outstanding approach. They demonstrate the linear and non-linear classifiers, also two Deep learning approaches. Logistic regression is used for creating the classifier, and deep learning is also used. Overcoming the Bag-Of-Word limitation, TF-IDF vector representation is used in the experiment. For getting the TF-IDF score, various types of grams methods are used especially unigrams, bigrams, and trigrams for the text. For deep learning, they experiment the 8 different types of neural networks including BIGRU-ATT, LW-HAN, X-BIGRU-LWAN, X-CNN-LWAN, MAX-HSS(Max poling over Hierarchical Attention Scores), and HAN. In their experiment, recall@k, precision@k, nDCG@k, and f1 score metrics are used. They get 0.698 scores for f1, and a 0.796 score for recall.

In the paper [8], MLP and CNN-based two methods are proposed by the authors for extreme multi-label text classification problems. In the experiment, the PubMed dataset is used. In the first experiment, word embedding is used for feature extracting and two dense layers are used for classification. One dense layer has 64 units and the last dense layer or output layer is used for prediction. Output layers units are the number of labels. In the second experiment, they just add the extra convolution layer after the embedding layer. For the convolution layer, the Kernel size is 5, and the number of filters is 100. Relu is the activation function, and reduction of the dimensionality max-pooling is used. The F1 score is

0.83, and 0.82 for CNN and Dense layer respectively.

In the paper [11], the Authors proposed CNN based model which is XML-CNN, also they compare this with 7 models that are based on the target embedding method, tree-based method, and Deep learning method. Their model is the improved version of the CNN-KIM architecture. In their model, they use word embedding for the representation of the words and after that convolutional layers with multiple filters that produce multiple features. Then dynamic max pooling is used. A fully connected layer is linked to the dynamic max pooling layers. Finally, a dense layer is connected to produce the output. For the final output, the sigmoid function is used to produce the sigmoid output. Six datasets are used for the experiment including Amazon-12k(12277 labels), Amazon-670k(670091 labels), wiki-500K1, wiki-30k, EUR-Lex (3865 labels), and RCV1 (103 labels). Model evaluation metrics are precision@k and Normalized Discounted Cumulative Gains(NDCG@K).

In the paper [20], A tree-based technique, AttentionXML, is proposed for extreme multi-label text classification. This experiment has two main features. Multi-label attention mechanism and the probabilistic label tree are those.

In the paper [3], A very popular target embedding method is proposed for extreme multi-label text classification. This method is called SLEEC. The Learning embedding and KNN makes the SLEEC architecture. The SLEEC has two steps of learning embedding. For the L-dimension label, this architecture learns L-dimension embedding that captures the non-linearity correlation from the distance metric. Using L-Dimension, KNN search projection in the prediction time. SLEEC also divides the training set for better relevance and fast search.

In the paper [15], The most popular tree-based model, FastXML, is proposed by the authors in the field of XMTC. FastXML has a great success in this field and high accuracy. This model learns from the hierarchy of training data. While training, it optimizes the NDCG for each node. A hyperplane parameterized has two subgroups and labels ranking. These two learn jointly. Documents having the same class are in the same node. In the prediction time, from root to leave node is considered and labels are added.

In the paper [10], the FastText method is proposed by the authors. FastText is the deep learning method. This method becomes popular because of its simple architecture. Though it is simple but effective. It follows the word embedding method for text representation. While making embedding, it averages the result. A softmax function is used for targeting the class labels. This word embedding method is the extended version of the CBOW [14].

In the paper [19], the author tries to implement the convolutional neural network on the text documents. This method is considered the earliest or first method of CNN. This method is known as CNN-Kim. All the versions so far we get are the upgraded version of this. Word embedding is applied in this method for word representation. Then Convolutional layer and max-pooling layer are linked. Finally, the fully connected layer is added for the output. This method is the base method of CNN.

In the paper [13], Bow-CNN is a CNN-based model. The full meaning of this method is Bag-of-word CNN. In this method, Bag-of-word is used for the word vector representation. Then convolutional layer and softmax output layer are added to the model.

In the paper [18], PD-Sparse is a different type of text classification method. The word procedure is different from other methods, this is why it does not belongs to the three categories that we know for extreme multilabel classification. This is a linear classifier. It learns from penalties based on the label matrix.

CHAPTER 4

Various Text Processing Techniques

In this modern era, We can not imagine our life without data. Data is everywhere including our mobile text, chat, medical sector, industry, and even the university library. We usually use raw data for our daily life. Almost all are unstructured, noisy, and raw data. In the natural language task, we get those unstructured data as input. We can not use those data directly to feed the machine learning algorithms, also all data are not useful or informative in real. Before providing the data into the machine learning algorithm, we must pre-process the data. Text or Data pre-processing is a technique that helps us to retrieve informative data from unstructured and noisy data. Also, processed data helps the machine learn better, and boost performance. The following sections go through the most common pre-processing techniques heavily used in natural language processing.

4.1 Standard Pre-Processing

4.1.1 Removing punctuations

In the raw text, it's common things to have punctuation like full stop(.), the comma(,), semicolon(;), percentage(%), dollar sign($), question mark(?), and so on. To better learn the model and performance, we need to remove all punctuation from the data.

4.1.2 Removing URLs

Nowadays, most of the data can be found on the internet. So, having URLs like "downloaded from https://www.github.com/mehedihasanraj" in the text data for reference is a common thing, but it is not informative for the machine learning algorithm. In the text preprocessing step, unnecessary URLs should be removed.

4.1.3 Changing the Case

"The" and "the" is the same word that we human know but the machine learning algorithm treats them as two different words. In the raw text, when it is a starting sentence, we generally find the first letter as a capital letter, but the same word we find in the middle of sentences as a small letter. To solve this problem, we need to transform our data into lowercase.

4.1.4 Removing the Stopwords

Stopwords are the words that occur frequently in the language like (of, and, too). Though they occur frequently of the language nature, they are not valuable for analysis. They have less meaning and sometimes no meaning in the machine learning model. We have to remove these stopwords. In Python, there is a library called nltk. We can use this library to remove the stopwords.

4.1.5 Remove words and digits containing digits

The mix of word and digit, or digit and digit is a normal thing in the raw data. For example, in the text, we sometimes use "2n8" for making understood as tonight or 5%. We get lots of words like that and those do not carry too much meaning. We have to remove those words that contain mixed of words and digits or single digits.

4.1.6 Removing the White space

In noisy data, we get many white spaces, and having many white space is bad for the machine learning model. To robust the performance of the machine learning model, we need to carefully remove the extra white spaces.

4.1.7 Spell Correction

Getting misspellings in unstructured data is frequent. Misspelled data is a barrier to the machine learning algorithm for better learning. In the pre-processing step, we have to carefully handle this problem.

4.1.8 Text Normalization

Text Normalization is the process of converting the text into a canonical form or representation. In social media posts, the texts are shortened or words are usually spelled in a different way. For example, on Facebook, or WhatsApp we sometimes write hello as "helloo", or bye as "byeee". For this type of problem, text normalization helps a lot. Text normalization makes the data into canonical form.

4.1.9 Stemming

Stemming is a text pre-processing technique that reduces the word to its root forms. It helps the words to be normalized. In the English world, there are many forms of a root word, base, or lemma. For doing stemming, a lookup table helps to make the word to be converted to its root or base form. That means there a are sets of rules in the stemming that help the words to be converted to its root form. A huge number of search engines use this for retrieving information. Stemming is important in the natural language task. As there are many variants available for the same word and they carry almost equal weight. For example, the word "play" has many variants like playing, played. All of these words carry almost equal information. So, we can use the root word for the information, or otherwise, our vocabulary will contain redundant words. To build a robust model, Stemming is important in the pre-processing techniques. In the research, we find several types of stem. One of them is "Porter Stemmer". This stemmer uses the five steps of the technique for word reduction. Another one is "Snowball Stemmer". Snowball Stemmer was created by Martin. It works more precisely and it is more logical with faster than the first one. The third one is "Lancaster Stemmer". Its word procedure is straightforward but sometimes it makes a meaningless word by doing over-stemming. The fourth one is "Regexp Stemmer". In this method, substrings are matched using regular expressions.

4.1.10 Lemmatization

In Natural Language Processing, Lemmatization is a very known word and one of the most common text pre-processing techniques. Lemmatization goals and Stemmer goals are the same and that is reducing the word to its root form. Lemmatization is a more advanced technique than Stemmer. The process we called Lemma. Though both's goal is the same, there are a few difference in the lemmatization and stemming. In the stemming, Stemmer chops off the trail or stems this. The stemmer has its own algorithm but while doing chopped, it does not know the meaning, it only knows how many characters should be chopped. But, in Lemmatization, this algorithm knows how much should chop to get a meaningful word. While Stemmer does not have this knowledge, Lammatizer has this knowledge, and this makes Lemmatizer more powerful. For example, The word "better". The base form of the word is good. Lemmatization knows that but Stemmer may chop this into like "bett" or "bet". So, using Stemmer may be bad for the model as we are losing valuable information and getting unwanted information. Though it is very useful in the natural language processing task, it has some disadvantages. Lemmatization is very costly in terms of time, though it is more accurate than Stemmer. If we use a chatbot or virtual assistant, Lemmatization is good as it has more meaningful answers.

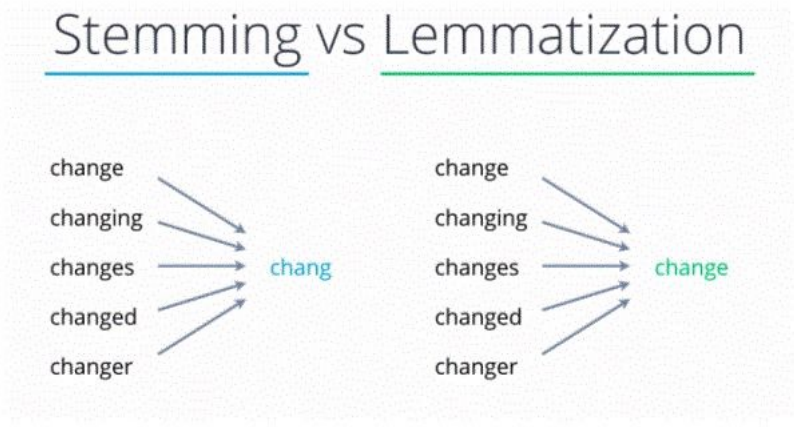

Figure 4.1: Stemming Vs Lemmatization

4.1.11 Tokenization

Tokenization is a technique that splits the data into pieces of meaningful phrases, sentences, words, or the entire text. These pieces are called tokens. Tokenization is a fundamental

thing in natural language processing and a difficult one. By using tokenization, our natural language processing model can express its behavior. Tokenizer helps to retrieve the meaning from the text. For example, the sentence " Deep Learning is important". If we tokenize this sentence and perform the most common tokenized based on space in a sentence, we get Deep-Learning-is-important. Why tokenization is important in Natural Language processing? Because it models the data, and also builds the vocabulary list. Tokenization is the building block of the vocabulary list.

Various Types of Tokenization

- Word Tokenization
- Character Tokenization
- Subword Tokenization
- Sentence Tokenization
- Byte Pair Encoding(BPE)

Word Tokenization

Word Tokenization is the common tokenization algorithm in NLP. In this tokenization method, sentences are split into pieces of words. For example, the sentence "Natural Language Processing has changed the world" is pieced into ["Natural", "Language"," Processing", "has", "changed", "the", "world"]. A common delimiter of this tokenizer is white space. The common problem of this tokenization is "OOV", also called "out of vocabulary". It fails to handle the outs of vocabulary words. But we can handle it by using unknown tokenization, but not so useful

Character Tokenization

Like word Tokenization, character tokenizer splits the text into some small pieces, but here the difference is that it does based on a set of characters. In every language, though words are many, characters are fixed. Because of this, the vocabulary in character tokenization is relatively smaller. In the word tokenization, there is a probability of having out of vocabulary. But in character tokenization, we can get rid of this problem. Example of character tokenization is like ["N", "a","t","u","r"," a","l"," ", "L","a","n","g","u"," a", "g", "e"," "," P","r","o","c", "e",

"s","s"," i","n","g"] for the sentence "Natural Language Processing". It can be different based on the condition. But using this tokenization, it is very difficult to find the relationship among the characters.

Subword Tokenization

Subword tokenization is another type of tokenization where each text is split into subwords or n-gram characters. In Character Tokenization, we find a difficulty and the subword Tokenization solves this issue. For example, "frequently" can be spit into ["frequent"," ly"].

Sentence tokenization

Sentence tokenization is another process to split the text into individual sentences. This tokenization breaks the huge text and makes its a set of sentences.

Byte Pair Encoding(BPE)

The another tokenization method is called Byte Pair Encoding also called BPE. This tokenization is used in the modern transformer-based model. In some cases, word and character tokenizers are not good like out-of-vocabulary (OOV) problems. The byte pair Encoding hands this problem very nicely. Byte Pair Encoding is a word segmentation algorithm. BPE marges the character or characters that are most frequent in the text. There are some steps to learning byte pair encoding. Firstly, it splits the words after appending </w>. Secondly, initialize the vocabulary. Thirdly, computes the pair of characters based on frequency. Fourthly, it marges the frequent pair. Fifthly, it stores the best pair, and it repeats the three and five numbers steps.

4.2 Features vectorization

After doing the standard pre-processing, Now it turns to represent the text into various representations

One-Hot Encoding(OHE)

In this technique, we first make the vocabulary for the unique token. The value zero and one are used for this token. One is used for the position and the rest of the place zero value is used. Though it is a very simple method, there are lots of issues with this method. In this method, the size of the unique word is the same as the vocabulary size, and the size of the vector for each word is equal to the vector size. This is very costly in terms of memory. Also, it does not capture the difference or informative information among different types of words.

Bag of Words (BoW)

A bag of words is an NLP technique that models the text like one hot Encoding. In another way, we can say that it is a method of extracting the features from the Text data. In the bag of words, the approach is very simple and flexible. It represents the occurrence of the word that is available in the input documents. The two basic things it does to extract the features. First, it makes the vocabulary of the known word, and the second step, measures or counts the presence of known words. In the One-hot Encoding representation, this only count the word, and put the value 1 if present otherwise 0. But, in the Bag of Words, it counts the frequency which means how many time one word is present in the sentence. So, the vector of represent value can be more than 0 to n.

Table 4.1: Bag of Word

Documents	the	cat	sat	in	hat	with
the cat sat	1	1	1	0	0	0
the cat sat in the hat	2	1	1	1	1	0
the cat with the hat	2	1	0	0	1	1

In the table 4.1, we can see that, first, it builds the vocabulary from the input text. After that, it makes the vector for each sentence and counts the frequency of the presence of each word in the sentences. In the second, in the sentence "the cat sat in the hat", the frequency count of "the" is 2 and other words are 1 except with. This is how a Bag of words is created. Though a bag of words is simple and more informative than one hot encoding representation, there are some limitations of the bag of the words. The first limitation is vocabulary building. If vocabulary is not created properly, then there is a high probability to make the sparse

matrics of the documents. The second limitation is sparsity. If the representations are large in terms of sparsity, then it is very costly and hard to model. Another limitation is meaning. It can not capture the meaning of the word as it is not preserving the order of the word. There is no semantic meaning to the sentence here.

Term Frequency-Inverse Document Frequency (TF-IDF)

While studying the Bag of Words, we find some limitations of the bag of Words. The most important issue is Bag of words does not contain information for the important words. To get rid of this issue, we get a new feature-extracting technique which is called Term Frequency-Inverse Document Frequency (TF-IDF). This is a scoring method that is widely used. It focuses on the document and how the document is related and relevant. This method is a statistical method. The two main word procedures of this method. The first one is TF and the second one is IDF. There are two metrics TF and IDF used for this method. Finally, mathematically it multiplies the two,

$$TF - IDF = TF * IDF \tag{4.2.1}$$

The first term TF represents the word frequency in the documents, and this frequency We can find using the bag of words technique as the bag of words calculates the frequency of the word. For example, We have three sentences that are shown in table 4.2

Table 4.2: Sentences

Sentence list
I love ML.
I love DL.
I love NLP.

If we pre-process the text including removing stopwords, punctuation, and converting to lower-case then we will get three modified sentences "love ml", "love dl", and "love nlp". If we count the word frequency then we get the table 4.3.

We get the frequency. Now we can calculate the TF or term frequency using the formula

$$TermFrequency = \frac{\text{Number of Repetition word in a sentence}}{(\text{Total Number of Word in a Sentence})} \tag{4.2.2}$$

Table 4.3: word frequency counting

words	Frequency
love	3
dl	1
ml	1
nlp	1

Table 4.4: Term Frequency counting

Term Frequency(TF)			
Sentence words	love ml	love dp	love nlp
love	1/2	1/2	1/2
ml	1/2	0/2	0/2
dp	0/2	1/2	0/2
nlp	0/2	0/2	1/2

After that we will calculate the Inverse Documents Frequency(IDF). For calculating this we use the following formula

$$IDF = log(\frac{\text{Number of Repetition word in a sentence}}{\text{(Total Number of Word in a Sentence)}}) \tag{4.2.3}$$

Then we will get the following table 4.5

Table 4.5: IDF counting

Words	Inverse Document Frequency	Total
love	log(3/3)	0
ml	log(3/1)	0.477
dl	log(3/1)	0.477
nlp	log(3/1)	0.47

As we know that TF-IDF = TF * IDF that means just multiply the two metrics, so we get the TF-IDF result for the three sentences table 4.6.

If we carefully see this, the "ml" word is more important in the sentence 1, "dl" is more important in the second sentence, and "nlp" is more important in the third sentence. The

Table 4.6: TF-IDF Result

	Feature no 1	**Feature no 2**	**Feature no 3**	**Feature no 4**
Sentences	love	ml	dl	nlp
love ml	0	0.238	0	0
love dl	0	0	0.2385	0
love nlp	0	0	0	0.2385

word "love" is not much important as it is available in all the sentences. So, TF-IDF maintains the word importance.

Word Embeddings

Word Embedding is a very important term while solving the NLP task. It is a modern natural language technique to extract features from the text. It is an advanced method and this method has the ability to understand the meaning of text-based content better than ever. So far, we have seen many techniques for feature extraction or feature representation. All have some problems when it comes dealing with semantics and context. All methods are not effective to capture the meaning of context, but the word embedding method solves this issue in modern NLP tasks. Word Embedding is a vector representation method, that helps to represent the word with the same representation that has a similar meaning. For example, the word football and cricket are almost similar meaning and their representation will be similar. There are some advantages to the word embedding method.

Advantages

- Reduces dimensionality
- Captures the semantic meaning
- Reduces the parameters
- Get idea among the words

Different types of word embedding techniques

- Word2Vec
- GloVe

- Bert Encoder

Word2Vec: Word2vec is one of the popular word embedding techniques in natural language processing. It has a great success. Word2Vec first proposed Google in 2013. In this method, words are represented as vectors, and words are closer if the words are similar otherwise words are far from. There are two major types of word2vec. One is CBOW or continuous Bag-of-Word, and another is continuous skip-gram. Both architectures are vice versa. In the CBOW, the current words are predicted based on the context. While Skip-Gram works based on the surrounding word.

GloVe: Global Vector for Word Representation or GloVe in another type of word embedding technique. The GloVe is the upgraded version of the word2vec method. By using the word occurrence matrix, it captures the global information. Word2vector captures the information from the local neighbors, but GloVe considers the whole. In entity recognition and word analogy, GloVe has had a great success.

FastText: FastText is also an extension of the word2vec method. Facebook proposed this method for word embedding. Other methods of representation are based on words, while FastText uses the n-gram characters for the representation. For example, the word "computer", represents [<co,omp,mpu,put,ute,ter,er>] if the n = 3. Where "<" indicates starting of the word, and ">" ending the word. By using this, it can capture the meaning of small words, prefixes, and suffixes.

CHAPTER 5

Extreme Multi-label Text Classification

Extreme multi-label text classification is different from other classification tasks like binary classification or multi-class classification. A huge number of the label makes the problem so difficult. Though it is too difficult, there has been significant progress made. Various methods have been proposed for this type of problem. If we categorize the proposed method, then we will get three categories. Target-Embedding method, tree-based method, and Deep learning methods are these. Most of the proposed methods are belongs to target-embedding and tree-based methods.

- Target embedding method
- Tree-based ensemble method
- Deep learning method

5.1 Categories of Extreme Multi-label Text Classification

5.1.1 General Perspective

Target-Embedding Method

In the Target-embedding method, the main focus is on the target value. Solving the data sparsity issue is by finding the low dimensional embedding of the labels vector. We find the

target-embedding method in the following proposal [2, 4, 6]. One of the methods is SLEEC belongs to this category. Another category is called Tree based method.

Tree-based Ensemble method

Three-based method is another type of category. In recent year, there are various types of tree-based methods has improved. In the paper [1, 16, 17], they have proposed several methods for the Tree-based Ensemble method. This method is similar to tree-based learning but the only difference is that it partitions each node in a recursive manner and each node contains a classifier. The tree-based model performs better for extreme multi-label classification.

5.2 Machine Learning Perspective

In machine learning perspective, we can divide Extreme Multi-label Text Classification into two categories.

- Traditional Machine Learning Method
- Deep learning method

5.2.1 Traditional Machine Learning

Traditional machine learning also dominates the text classification problem. Traditional machine learning has three main learning methods. One is a probability-based machine learning method, the second one is a neighbor-based machine learning method, and the final one is the tree-based method.

- Probability based
- Neighbor-based
- Tree based

A recent probability-based machine learning algorithm is Naive Bayes(NB). NB performs well, and K-nearest neighbor or K-NN is another type of machine learning algorithm based on neighbor-based and used for text classification. It gives the result based on the neighbor's state. The decision tree is another type of machine learning algorithm and is popular in

text classification. The decision tree is the tree-based algorithm. If we talk about another successful machine learning algorithm, then the Support vector machine comes. Support vector machine is very popular when the dimension is higher. SVM predicts the probability for the class. The logistic regression algorithm is the earliest algorithm that is used for text classification. It also predicts the probability over the class.

Table 5.1: Traditional Machine Learning

Advantages	**Drawbacks**
Only needs less amount of data	Failed at large dataset
Very Cost effective	Feature engineering needed
Success at Rule based approach	Failed to capture complex patterns

5.2.2 Deep Learning

Deep Learning method has the significant success of the text classification, and This is why we will discuss this method in details. Deep learning term is common in this modern era. Deep learning is a modern technique that enables computers to perform tasks that comes naturally to human. Deep learning is a subset of machine learning. Deep learning learns by example. A few decades ago, what is impossible to do by computer like text classification, image processing, speech recognition, and so on, now that has been possible through deep learning. Deep learning learns how a human learns. In traditional machine learning, we have to extract the features from data and then we can train the model. Extracting features is very difficult work, especially when there is huge data, also some current data like image data, speech data, and text data. It is very difficult to select the features from this type of data. But, deep learning makes the work so easy, as it can automatically do feature extraction and modeling, and also it has higher accuracy for extracting the features than humans. There are three major types of Deep learning, and they are.

1. Multilayer Perceptron (MLP)
2. Convolution Neural Network(CNN)
3. Recurrent Neural Network(RNN)

These types of Deep learning are now often using in text classification

Multilayer Perceptron

Multilayer Perceptron is nothing but two or more layers having one or more perceptron. A perceptron is a single-neuron model. Neuron term comes from the biological brain that is used for solving difficult tasks. Perceptron is a building block of the artificial neural network. Perceptron is a mixture of 4 parameters, and they are input value, weights, bias, and activation function.

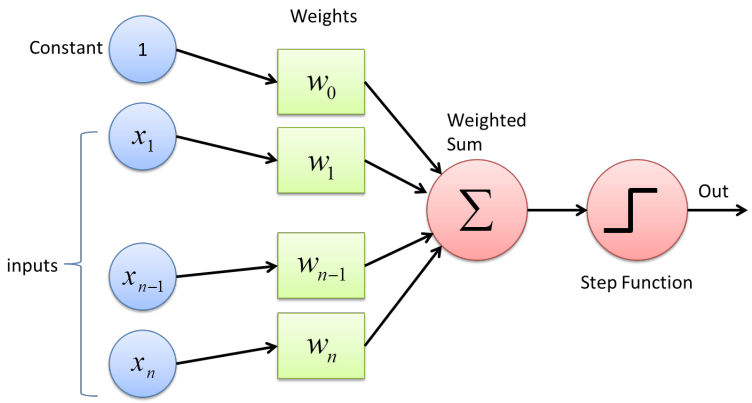

Figure 5.1: Perceptron

In the picture 5.1, we can see a simple perceptron. The output of the simple neuron is produced through some mathematical calculations. In the perceptron, there are some inputs, the input is multiplied by some weights. Weights are nothing but initialized random values. Weight is an important part of the perceptron as it learns from the example. After that, there is a bias that is added to the weight. Then the sum is passed through an activation function and produces the output.

$$Output = ActivationFunction\left(\sum(((input * weight) + bias))\right) \tag{5.2.1}$$

Activation Function: The activation function activates the neuron of the neural network. The transfer function is another name for the activation function. The activation function has a great impact on neural network performance, and we use different activation functions for different parts of the neural network. In deep learning, there are three types of layers. One is the input layer that takes raw input, another is called the hidden layer that takes input from another layer, and finally, we get output layers. In every layer, we may use different types of activation function.All hidden layers use the same activation function for each perceptron in the hidden layer, and the output layer uses another type of activation function that depends

on the prediction type, for example, if we need binary prediction we may use the sigmoid function, if we need probability prediction, we may use the softmax function. There are two major types of activation function, one is the linear activation function, and another is the non-linear activation function.

Linear Activation Function The linear activation function takes the input and produces the output. This output can be any range between negative infinity to infinity.

$$f(x) = x \tag{5.2.2}$$

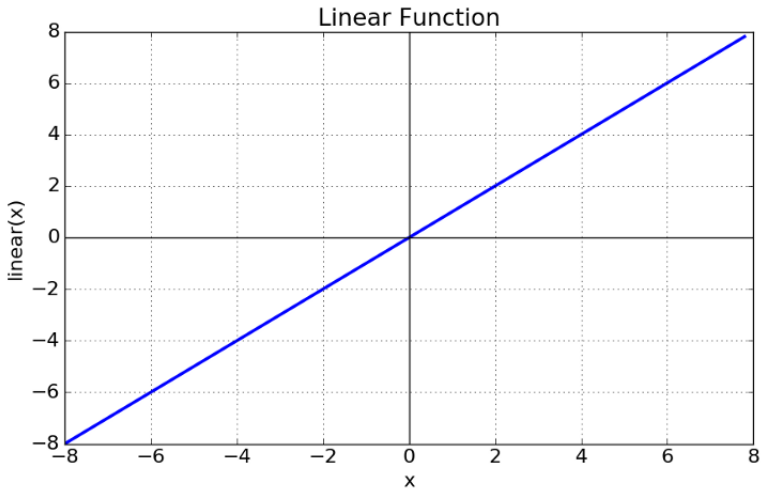

Figure 5.2: Linear Activation

Linear activation function is not useful and it does not help the model as its input and output is the same. It can not capture the complex pattern.Only this is the linear activation function, rest of them are non-linear.Non-Linear activation function is important in deep learning. This activation function is used widely. The non-linear activation function helps the model to capture the complex pattern.Most of the data are complex pattern and holds the non-linear pattern in deep learning. A non-linear activation function can capture this. There are lots of Non-linear activation functions, and they are discussing below.

Sigmoid or Logistic Activation Function: Sigmoid function is a widely used activation function. It follows the probabilistic method and makes the output range [0 to 1].

$$f(x) = x \tag{5.2.3}$$

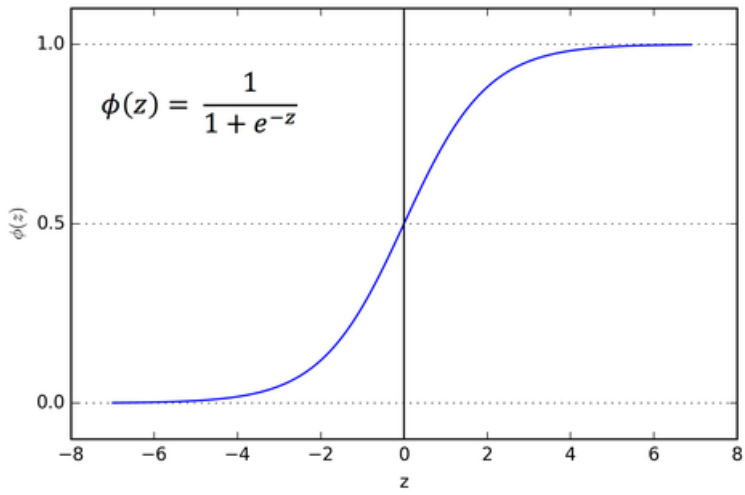

Figure 5.3: Sigmoid activation

It can convert any input to a range between 0 to 1. The mathematical equation of the sigmoid function is

$$f(x) = \frac{1}{(1 + e^{-x})} \tag{5.2.4}$$

This activation function is useful while making decisions for binary output, also it is useful for multilabel classification problems, and for this, it is used in the output layer. The sigmoid function has a problem of vanishing gradient problem. It makes the neural network get stuck while training.

Tanh or Hyperbolic Tangent Activation Function: Tanh or Hyperbolic Tangent function is another type of activation function, it is similar to the sigmoid function, but its range between -1 to 1. It is much better than the sigmoid function. In two class cases, this activation function is used mostly.

$$f(x) = \frac{(e^x - e^x)}{(e^x + e^x)} \tag{5.2.5}$$

The limitation of this function is, the learning rate is slow if the model uses this activation problem.

ReLu or Rectified Linear Unite: In deep learning, the ReLu activation function is a common and mostly used function right now. It is used in almost all CNN problems or Deep learning problems. ReLu activation function range between zero to infinity. The mathematical equation of the Relu is.

$$R(z) = max(0, z) \tag{5.2.6}$$

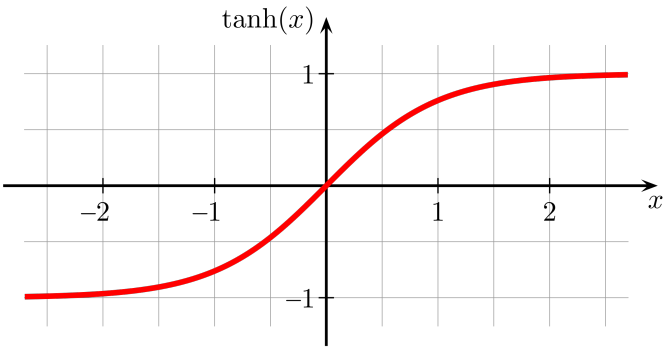

Figure 5.4: Tanh activation

So, this function converts all negative inputs to zero and all positive inputs remain the same as the input. It learns fast so fast than the sigmoid function and tanh function. It learns too fast because it converts the negative value to zero. Sometimes, it creates problems as data is not fitting properly because of that.

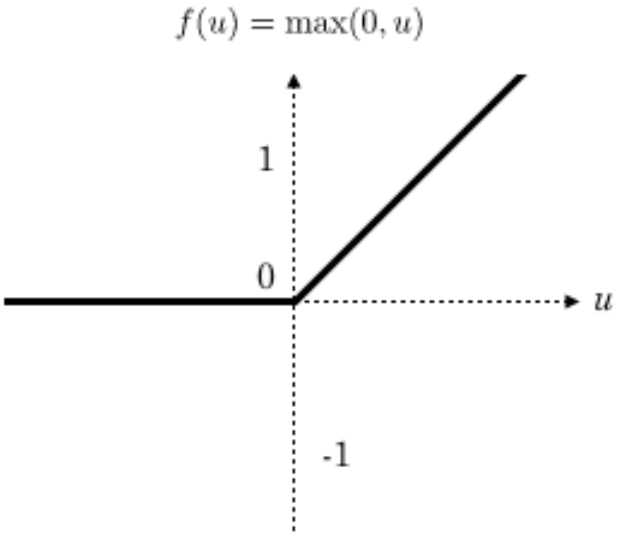

Figure 5.5: ReLu activation

Leaky Relu: Leaky Relu is an updated version of the Relu function. It solves ReLu's "Dying ReLu" problem. Unlike Relu, Leaky Relu does not make the zero for all the negative input, instead, it assigns a value that is near zero. By using this technique, Leaky ReLu can solve the ReLu problem

Softmax Function: Softmax is another type of activation function that is used mostly in the last layer or output layer. It produces the output like the sigmoid function. It takes the input vector of raw output from the neural network and Softmax gives the probabilistic score

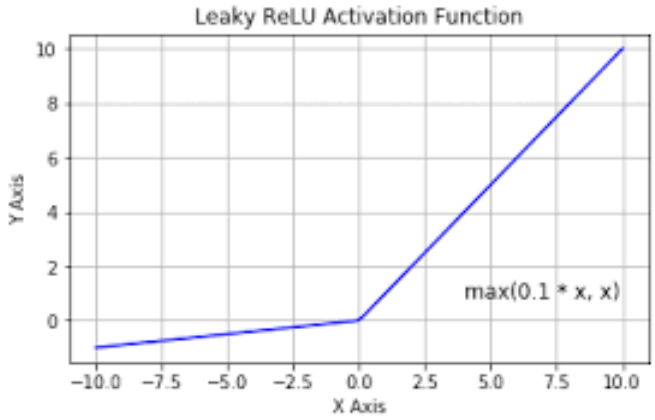

Figure 5.6: Leaky ReLu activation

as output. The mathematical equation is,

$$softmax\left(z\right)_i = \frac{e^{z_i}}{\sum_{j=1}^{N} e^{z_j}} \tag{5.2.7}$$

Where, z = input from Neural network
E = exponential,
I = ith entry in the softmax output vector

Softmax function is important for multiclass classification, also we can softmax function for the binary classification problem.

Convolutional Neural Network (CNN)

Convolutional neural network is a deep learning architecture and is well-known in the field of image, and audio processing. CNN is most successful in the computer vision sector, but it also can be used in various sectors that we shown in our experiment. There are much more things that make convolutional neural networks the most popular. There are various types of CNN architecture are available, like AlexNet, VGGNet, GoogleNet, ResNet, and so on.

Advantages of CNN

- No need the manual feature extraction, and it can learn directly from data.
- It has high accuracy achievement.
- It can be used for new tasks by retrained
- It shares the weight and bias, so get fewer parameters

Convolutional Neural Network has involved three major layers

- Convolutional
- Pooling
- FC or Fully Connected layer.

Among these layers, the Convolutional layer is the first layer, after that, we use the pooling layer, and finally, the Fully Connected layer.

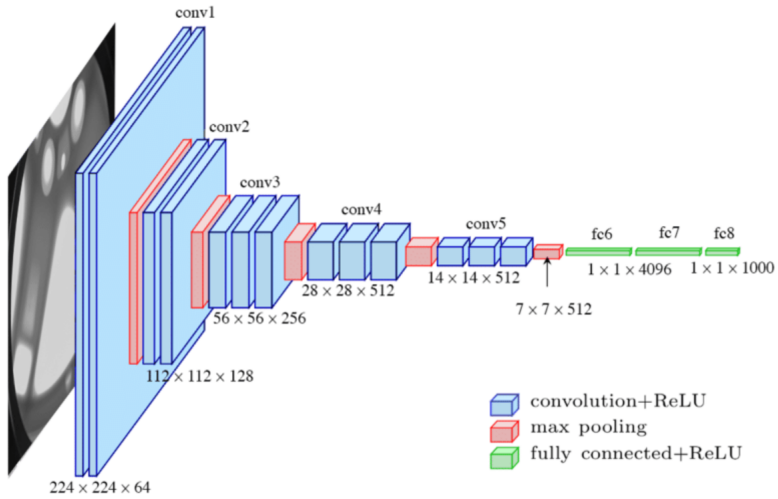

Figure 5.7: CNN Model

Convolutional Layer: In the CNN, the convolutional layer is the most important part, and most of the computations occur here.In this part, there are a few parameters involved, like the number of filters, filter size, stride, and padding.

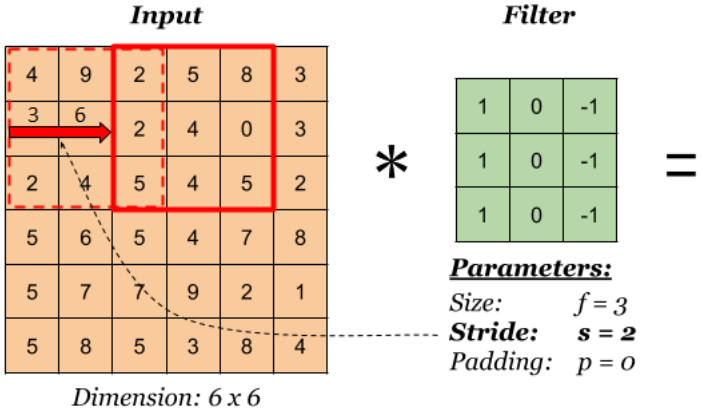

Figure 5.8: Convolutional

Filter or kernel: Filter or kernel is an important term in the convolutional layer, filter affects the output depth, and it helps to learn the features. Filter size can be N*N dimension, where N = 1,2,3,....

Stride: Stride is nothing but how many steps the kernel will move. The stride value usually belongs to 1. But more than one can be possible, and this is rare. If the value of the stride is large, the output becomes small.

Padding: When the input is not fitted for the kernel, that time padding is used. Padding allows for covering more space for the kernel in the input data. In the upper corner, and lower corner side data are not covered properly while the kernel moves. To solve this problem padding is used. There are three main types of padding. The first one is valid padding which is called no padding. the second one is the same padding, it allows us to keep the output size the same as the input size. The last one is full padding which increases the output size by adding zero.

Pooling Layer: Another important layer is the pooling layer. The pooling layer is used for reducing the number of parameters in the input. By moving the kernel in the entire input, and putting the value that we want. In general, we get two types of pooling, one is max pooling and another is average pooing. Max pooling is used to put the max output from the input when the kernel moves, and average pooling is the same as max pooling but put the average output. It helps to solve the overfitting problem.

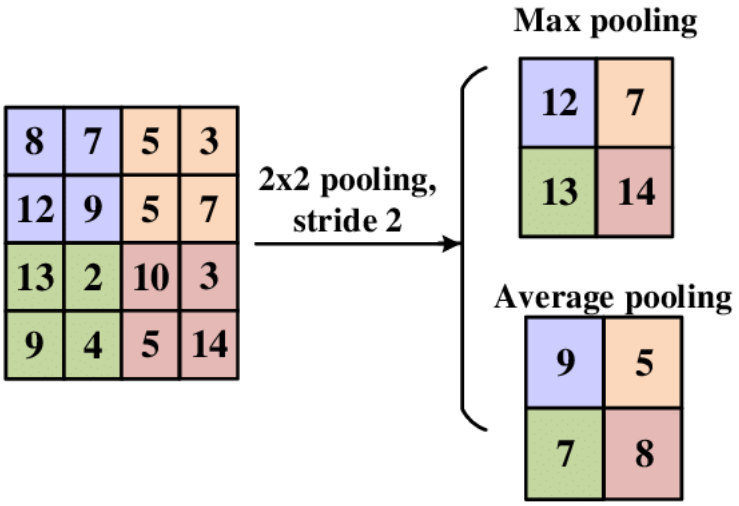

Figure 5.9: Pooling

Fully Connected layer: For CNN based model, the fully connected layer is the last layer. A

fully connected layer takes the input from the previous layer, and it is used for the prediction or output.

Recurrent Neural Network (RNN)

Recurrent Neural Network is another type of neural network. A recurrent neural network is best for sequential data. Many studies show that in time series problems, RNN performs outstanding. RNN dominates areas like machine translation, speech recognition, video tagging, text summarization, language modeling, and so on. RNN is popular, because it has some benefits, like proving the same weight and bias to all the layers that solve the huge parameter problem, also its memory cell gives the ability to store the information and help to perform better.

Advantages of RNN

- Can process long input length
- Model size does not depend on input size.
- Store important information
- Weights shares

Drawbacks of RNN

- Slow computation
- Difficult to store information for a long time

Variety of RNN

1. Simple RNN
2. Gated Recurrent Unite(GRU)
3. Long Short Term Memory(LSTM)

Simple RNN: Simple RNN is the simplest type of RNN and this the first type of RNN. The "simple" term means in the network, the cell of the RNN is not modified. Fully connected

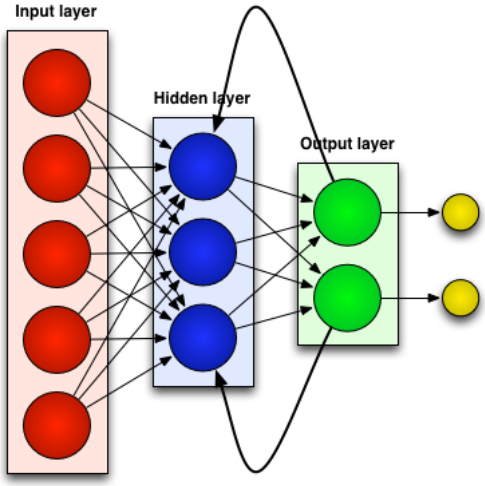

Figure 5.10: Simple RNN

networks are the base architecture of this variety. There is also a feedback loop in the neuron. These feedback loops are important for future prediction.

Long-Term Short-Term Memory(LSTM): LSTM is another variety of RNNs. simple RNN can not process long sequences. LSTM overcomes this problem. LSTM is good at processing long sequence data. The structure of the LSTM is slightly more complex than simple RNN. In the simple RNN, to obtain a new state, it takes input at the time stamp and the hidden state, but LSTM takes input from three different states. One is from the input stage, the second is from short-term memory, and lastly from long-term memory. LSTM consist of three types of gate. They are the input gate, forget gate, and output gate.

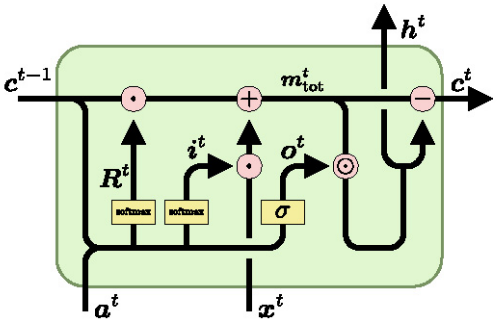

Figure 5.11: LSTM Architecture

Input Gate: The input gate is a gate that filters out unnecessary data, and it does this filtering using current input and the previous step or short-term memory. The input gate is responsible for where information should be stored or not.

Forget Gate: Forget gate is used for discarding information from long-term memory. It uses the two vectors for the work. Forget vectors and long-term memory vectors are these two vectors.

Output Gate: The output gate is the gate that produces the new short-term memory. For doing this, the output gate uses the current input, the latest short-term memory, and the new long-term memory.

Gated Recurrent Unite(GRU): Gated Recurrent Unite or GRU is another variety of the Recurrent neural network. GRU mechanism is the same as simple RNN, but here the unique difference is GRU unite. In the simple RNN, there are some problems, and GRU overcomes the problem. LSTM uses three gates whereas GRU uses two gates, one is the update gate and another is the reset gate. GRU does not have the Output or memory unit like LSTM.

Update Gate Update gate is very important in the GRU architecture. This gate is the only one responsible for whether that information will pass in the next state. This helps the RRN to solve the vanishing gradient problem by copying information

Reset Gate: Reset gate is used for resetting the data or how much data is needed. It also decides whether previous data is important or not.

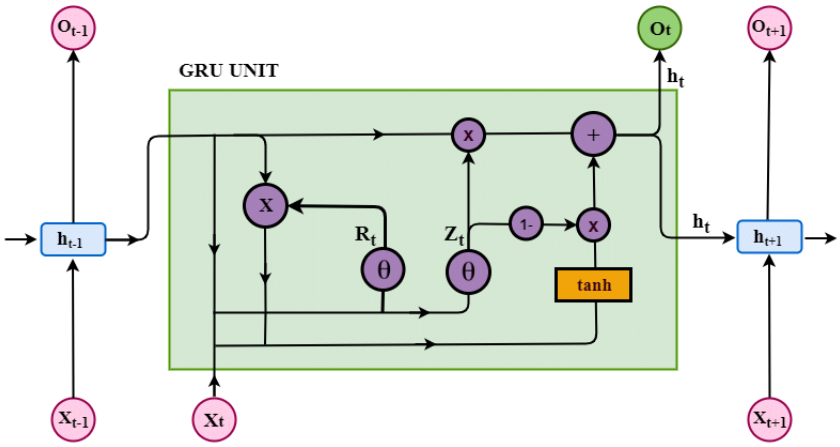

Figure 5.12: GRU Architecture

Table 5.2: Deep Learning

Advantages	Drawbacks
Can capture complex patterns	Lots of data needed
High performance	Coltly in terms of training
Less Feature Engineering	Costly for data collection

CHAPTER 6

Data-set Description

In this section, we are going to introduce the dataset, that we have used for our experiment. We have collected the dataset from Kaggle, and that is "Title Based Semantic Subject indexing". Kaggle is the most popular side for datasets. Semantic Subject indexing is, annotating texts with terms that define the content. It is frequently used in the online library system. Annotating mostly does the human expert.

6.1 Content

This dataset contains two files, one is "EconBiz" and another is "PubMed". In our experiment, we used the "EconBiz.csv" dataset.

6.1.1 EconBiz Dataset

"ZBW-Leibniz Information Center for Economics in July 2017" builds the dataset from a meta-data export. Dataset is tagged as English with STW marks and removed the duplicates by checking for titles and labels. Then there are 1065k publications. Dataset labels are chosen by expert human annotators from the Standard Thesaurus Wirtschaft(STW), which has over 6000 labels.

6.1.2 Attributes or Field

Dataset contains the following data field

- id
- title
- labels
- fold

In the "id" field, there is the unique identification for the publication. It is an integer value. the "title" field is nothing but the title of the publication. In the "labels" section, there is a huge number of labels are labeled as a string and separated by TAB. The "fold" field contains integer values. Folds are belongs to 0 to 9 values.

CHAPTER 7

Methodology

In this section, we will discuss the work methodology for our experiment. Our experiment is divided into some important stages.

- Pre-processing
- Model Building
- Training the Model
- Evaluation of the model with different Metrics

7.1 Pre-processing

We are solving the NLP problem. In the NLP task, pre-processing is the first thing for building a successful model. In the Natural language task, we get the raw data or noisy data. Raw data and noisy data are not useful for capturing the pattern. If we feed the noisy data, then model performance will be decreased. Model performance depends on how well data is pre-processed. We have used more than 500k data for our experiment so that we can compare the model performance with our referenced result. They use 500k data for their experiment. From our dataset, we have taken 500k data and saved it in another file, then we work on this dataset as we have limited resources like memory and ram. Title processing, Label processing, and Feature representation are the three main part in the pre-processing step.

7.1.1 Title Processing

The title contains lots of words with different cases. Some words are capitalized, some are not, and some are mixed. We make the all word in the title in lowercase. Lowercasing is important because the word "Market" and "market" are the same but computer treats both are different while doing tokenization. We use "lower()" function to make the title lower. We then try to remove the numbers from the title as numbers are not so much informative for the model performance. After that, we remove the punctuation and the single characters from the title. We remove the stop words. Stops words are the words that are common in the language and there is no semantic meaning, for example, 'the', 'am', 'is', and so on. These stopwords are not informatics for the features, this is why we are removing them. For this, we use the 'nltk' library. We also use the "wordtokenize()" function that returns the list. Then we use the lemmatize operation in the word because we may get the same word with different forms like "go", or "going". By lemmatize operation, we get the root form of the word and keep the root word. Then finally use the join operation on the processed list to make the sentence for each title.

7.1.2 Label Processing

In the dataset, labels are represented as strings. We can not use this for training the model. Before giving the labels to the training phase, we have to process them. In extreme multi-label text classification, there are lots of labels. In the string, labels are separated '/t' delimiter. We use the "split()" function to spit each label from the label string. As our labels are multi-label, so we need multilabel binarization representation. For this, we use the "MultiLabelBinarizer()" function. By using this function, we fit our labels, and training time we transform labels.

7.1.3 Feature Representation

So far, standard pre-processing steps are done. Now, we have to represent the text data into numeric form, because the computer can not understand and process the text data. For representing the text data into numeric form, we use the tokenization method. We did not fit the all word in the tokenizer method, instead filtering the words list first. In the filtering, we first count all word frequencies, then we only have those words, having more than two

frequencies, and also filter out those words having less than 2 lengths. We use our build in method for filtering. We do this, so that our vocabulary list doesn't become large. Then we finally fit those filtered words into the tokenizer. We also use the "text_to_sequence" method for getting the final representation of the feature. Different sentence length may be different and most of the deep learning algorithm takes the fixed length as a input, so padding is also added and the max length is chosen to 100. For the padding type, we use the "post" padding.

7.2 Model Building

In this section, we will explain how we build our models and what are the parameters. For our experiment, we build 4 models. One is CNN based model, the second one is MLP based model, the third one is GRU based, and the final one is LSTM based model. In every model, we use the embedding layer from the Keras library for features representation. In the embedding layers, we use the input length as 100, the input dimension size is equal to the size of the vocabulary, and the output dimension size is 300. It is noticeable that we use keras auto-tuner for tuning our parameter of every model, that is not added the code for becoming messy.

7.2.1 CNN model

In our CNN model, We use the Conv1D from the Keras library. For the parameter, the filter size of 600, where the kernel size is equal to 3 is used. For the padding, we use the "valid" padding. For kernel moving step size or strides, we use the value 1. "ReLu" is the activation function for the Conv1D. After that, we added a pooling layer to reduce the model parameter. "GlobalMaxPooling1D" is used for the pooling. A dense layer with 1024 units is added to capture the complex pattern, and "ReLu" is used for the activation function. In the final layer or output layer, we add a Dense layer. The unit number is equal to the total number of unique labels in the dataset. The "Sigmoid" activation function is used for the activation. For the multi-label classification problem, the Sigmoid activation function is the best choice. "binary_crossentropy" is used for loss, and "adam" is used for optimizer with a 0.001 learning rate.

7.2.2 MLP model

In our MLP, after the embedding layer, we use the "Flatten" layer to flat the embedding output as the embedding output is not one dimensional. The flatting layer reshapes the output. Then we use the Dense layer with 1024 units and "ReLu" is used for the activation function. We link this layer to the final Dense layer whereas Dense unit size equal to the number of prediction labels. Like the CNN model, we use the same loss function which is "binary_crossentropy", and the same optimizer(Adam) with the same learning rate.

7.2.3 GRU Model

We use the same parameter for embedding layer in the GRU model. Then we use a dropout layer with a value of 0.01. A GRU layer with unit 64 is added to the model. This GRU layer is bidirectional. In the model, A dense layer with 600 units, and a dropout with a value of 0.01 is also added. ReLu is the activation function. Finally, a dense layer with a number of units equal to the number of prediction labels is added. For the activation function, sigmoid is used, "binary_crossentropy" is the loss function, and "Adam" is the optimizer.

7.2.4 LSTM Model

In the LSTM model, initially we use the same embedding format. A dropout layer is connected with the Embedding layer with a 0.01 value. Then we add the Bidirectional LSTM with a unit size of 64. We add our LSTM output with another Dense layer and dropout layer, where Unit 600 is for the dense unit, and 0.01 is for the dropout layer. Activation function relu is used. Then finally, we use the same output layer that we use for CNN and MLP models. We also use the same loss function and the same optimizer.

7.3 Training the Model

In this step, we will explain how we train the model. We first split out 500k data into train and test part that is 90% and 10%. We use 10% for testing because our dataset is imbalanced due to extreme multi-label text classification, we try to give the data as much as possible in the training phase. For splitting the dataset, we use the "train_test_split" function. We only

use the training dataset for training the model, and the test dataset is used for testing the model. For training, the "fit" function is used. We use the 7 iterations for training the CNN model otherwise it is going to be overfitted, We use 10 iterations for the MLP, and 12 for both GRU and LSTM.Where the batch size is used 128. For each iteration, we predict the model output using the "model. predict()" function. In the model's last layer, we use the sigmoid function, so we get the probability as output. We use the different threshold values of [0.20, 0.25, 0.30] for getting the model score. We use "f1score()" function to get the F1 scores, also model gives the score for the default threshold value of 0.50.

7.4 Evaluation of the model with different Metrics

In this section, we will discuss different types of metrics that are mostly used in the classification task, particularly, in the extreme multi-label text classifications.

7.4.1 Different types of classification metrics

Confusion matrix

Confusion Matrix is a performance measurement metric for machine learning classification problems, Also another way we can say that it is the summary of the prediction result for the classification problem. The confusion matrix is a common term in machine learning while we deal with the classification problem. It is a crucial thing in the classification task, because it gives information on errors that the classification models are doing, and also it gives the information on which type of errors is happening. The confusion matrix is extremely needed for various types of measurement metrics like precision, recall, accuracy, specificity, and so on. A confusion matrix is an N X N matrix where N is the target class. If the target class is 3 then the confusion matrix will be a 3 X 3 matrix. Here is an example of only a 2 X 2 confusion matrix.

In the confusion matrix, we generally find 4 terms. TP, FP, FN, and TN. Where TP means True Positive, FP means False Positive, FN means False Negative, and TN means True Negative. We also can get a better understanding of the model by looking at the confusion matrix table. A good model has a higher True Positive(TP), and True Negative (TN). False Positive (FP) defines the type 1 error, and False Negative (FN) defines the type 2 error.

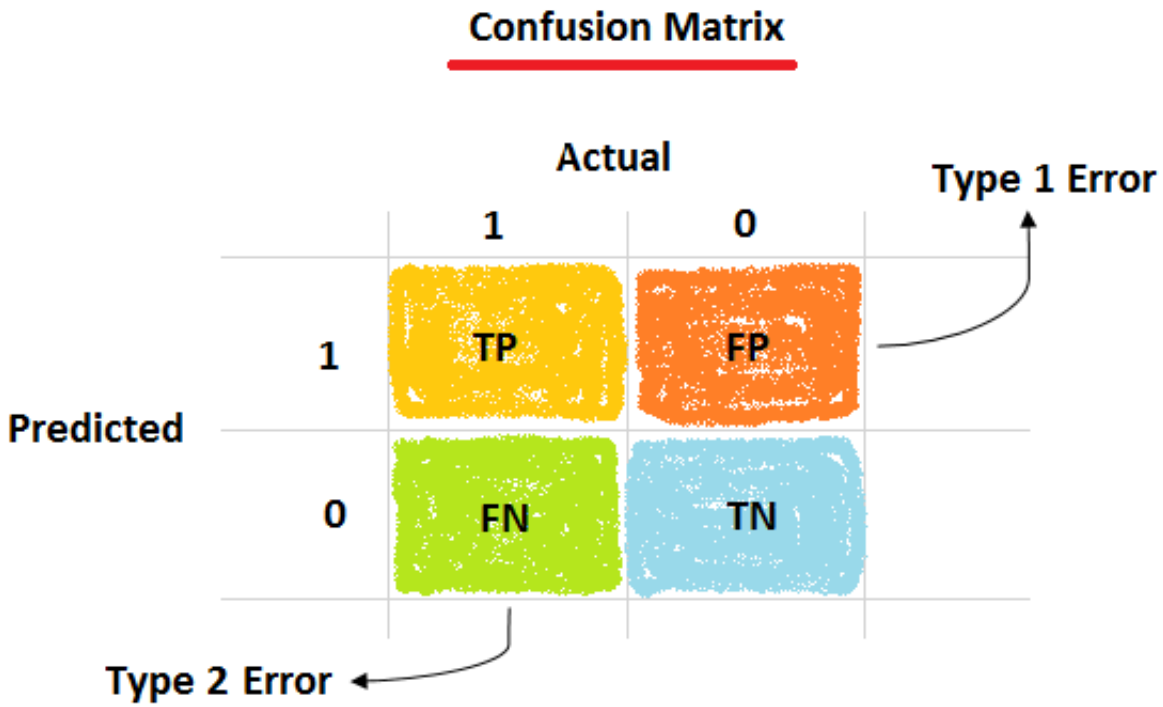

Figure 7.1: Confusion Matrix

True Positive(TP): Total number of values when predicted and actual both values are positive

True Negative(TN): Total number of values when predicted and actual both value is negative.

False Positive(FP): Total number of values when predicted is positive but actual is negative.

False Negative(FN): Total number of values when predicted is negative but the actual is positive.

The confusion Matrix gives a better idea of the Model. We can get this model information after looking at the confusion matrix.

Precision

Precision is a machine learning performance metric. Precision is the simple ratio between the true positives and all the positives. Precision is important for visualizing the model's reliability. Precision can help the model for improving the model and where error is happening that can be identified. Precision is a popular metric in multi-label text classification problems. We get the precision value using the given formula.

$$Precision = \frac{Truepositive}{(TruePositive + FalsePositive)} \tag{7.4.1}$$

Recall

Recall is another performance metric for classification problems. It is not bound to binary classification, but it also uses for multi-label text classification. Recall gives the information about the correct positive prediction rate. Precision only gives the positive prediction rate among all positive predictions, while recall gives the missed positive prediction indication.The recall range can be 0 to 1. We can find the recall score using the following formula.

$$Recall = \frac{Truepositive}{(TruePositives + FalseNegative)} \tag{7.4.2}$$

F1 Score

The weighted average of precision and recall is the F1 score. Unlike, precision and recall only focus on positive or negative, the F1 score focus on both. Because of that, the F1 score is very popular in the imbalance dataset problems. Almost all extreme multi-label text classification problems are imbalanced, and F1 is a common measure in multi-label text classification. F1 can have a maximum value of 1 which means a perfect balance of precision and recall and a minimum of 0 which means either precision or recall is zero. It is a trade-off of precision and recalls. For our performance metric, we use the F1 score for model validation.

$$F_1\ Score = 2 * \frac{Precision * Recall}{(Precision + Recall)} \tag{7.4.3}$$

7.4.2 Model evaluation Metrics

As our problem is the extreme multi-label text classification problem, we use the f1 score for our model evaluation metric. We chose this metric because it is a very popular metric as it is the average of precision and recall. Many research use f1 as a model metric [7, 12].

CHAPTER 8

Result & Discussion

In this section, we are going to discuss our experimental result. Our Experimental result are divided into some categories.

8.1 Training and Test Loss Result

Models main goal is to keep the loss as low as possible. When said a model is good when it has comparatively low loss in both training and testing. But both training and test loss should have a great trade-off. In the plot 8.1, we see the training and test loss for different type of models. In the CNN, we can see that at the beginning of the epoch, training and test loss is high as usual. When the epoch number increases, training loss is decreasing continuously, and test loss is also decreasing. However, after 5 epochs, training loss is decreasing, where test loss is increasing. So, the model is going to be overfitted. So, we stop the training. Between 2 to 5 epochs are the best trade-off for training and test loss of the CNN model. It is noticeable that we use the 0-based epoch. The test loss of the CNN model is between 0.003 to 0.004. MLP model result is similar to the CNN model result. MLP models loss is decreasing according to epoch numbers. But, after a certain epoch, test loss is increasing instead of decreases. 3 to 5 is the best epoch for the MLP model loss. In the GRU model, After 6 epochs training loss is decreasing but the test is not. So, for the GRU model, test loss is near to CNN with epoch range 6 to 9. Same as GRU, the test loss is between 0.003 to 0.004 with an epoch range of 7 to 10 being the best test loss for the LSTM model.

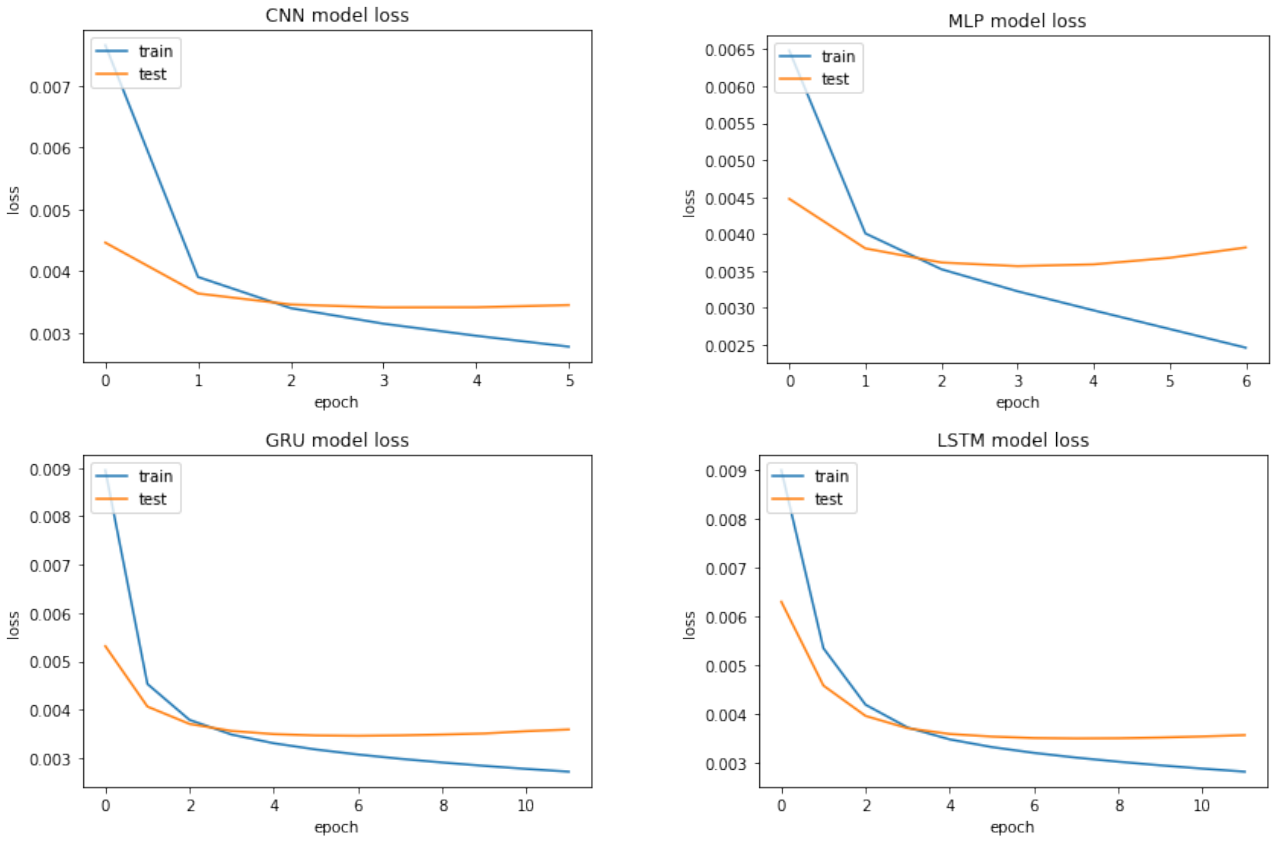

Figure 8.1: Different Model Loss

8.2 F1 Result with Different Threshold

In the given plot 8.2, we get the f1 score with a different threshold value. The F1 score is calculated in each epoch with 0.20, 0.25, 0.30 and a default 0.50 threshold value. For an extreme multi-label text classification problem, the default threshold is not good as we found in the different studies. In the paper [1], they use the threshold value 0.20 instead of the default 0.50 and get significant results. In the plot for CNN, MLP, GRU, and LSTM, we see that in the beginning, 0.20 is the best threshold and gives the best f1 score, but when the epoch increases 0.20 becomes weak compared to the 0.25 threshold value. For better results, 0.25 is the best threshold value that we find in the final. It is noticeable that the default threshold value always performs bad compare to others.

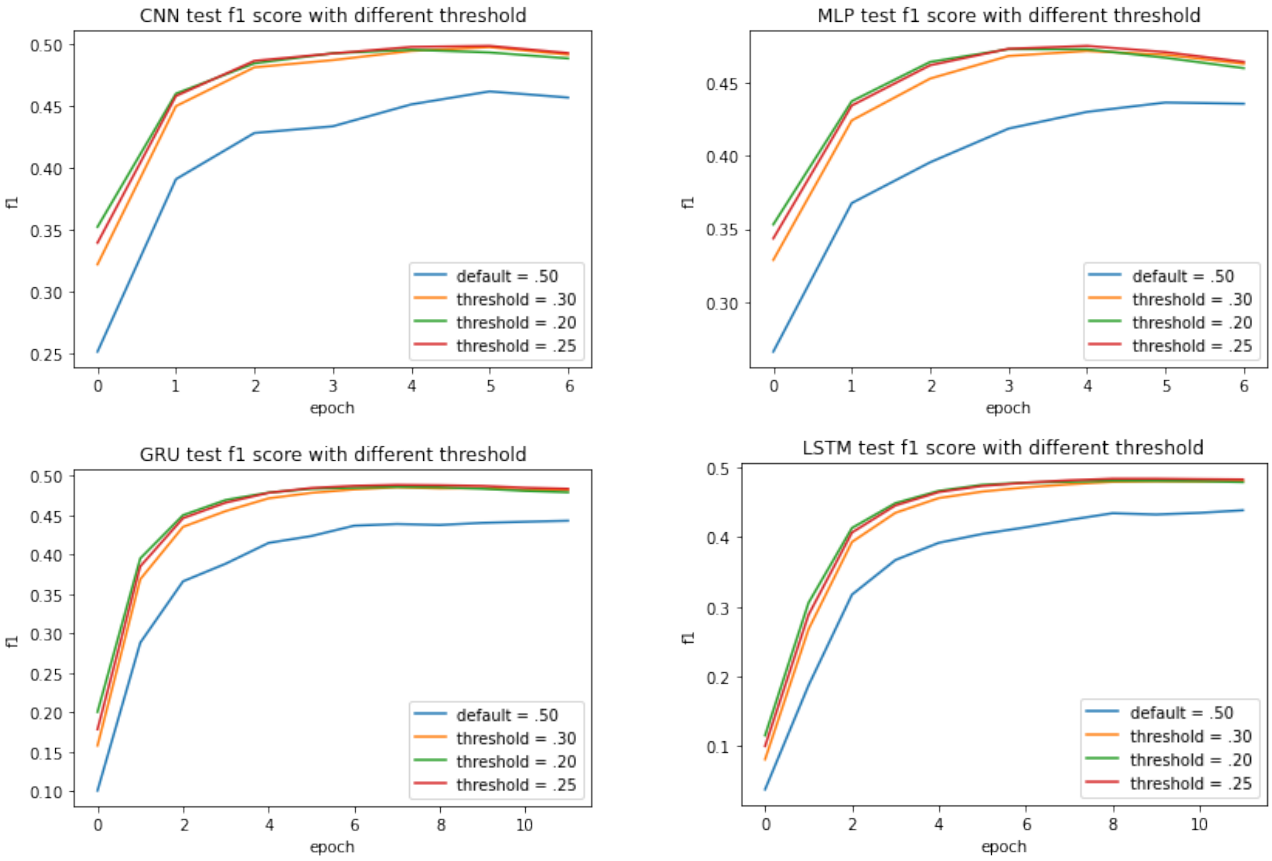

Figure 8.2: Model Loss with different threshold value

8.3 Training and Test F1 Results

Given plots 8.3 gives the comparative result for training and testing f1 scores with epochs number. In the CNN model, test f1 increases with training f1 for higher epoch numbers, but between 4 to 6 epochs, test f1 score is not increasing while training f1 score is increasing. So, after that, it's going to be overfitted between 4 to 6 epochs, the CNN model gives the best result, particularly at 5 epochs, CNN gives the best f1 score. In the MLP model, like the CNN model, the same thing happens between 3 to 5 epochs, MLP gives its best test f1 scores. In the GRU model, between 7 to 10 epochs, test f1 scores are the best scores for this model. Model LSTM gives the best test f1 between 7 to 11 epochs

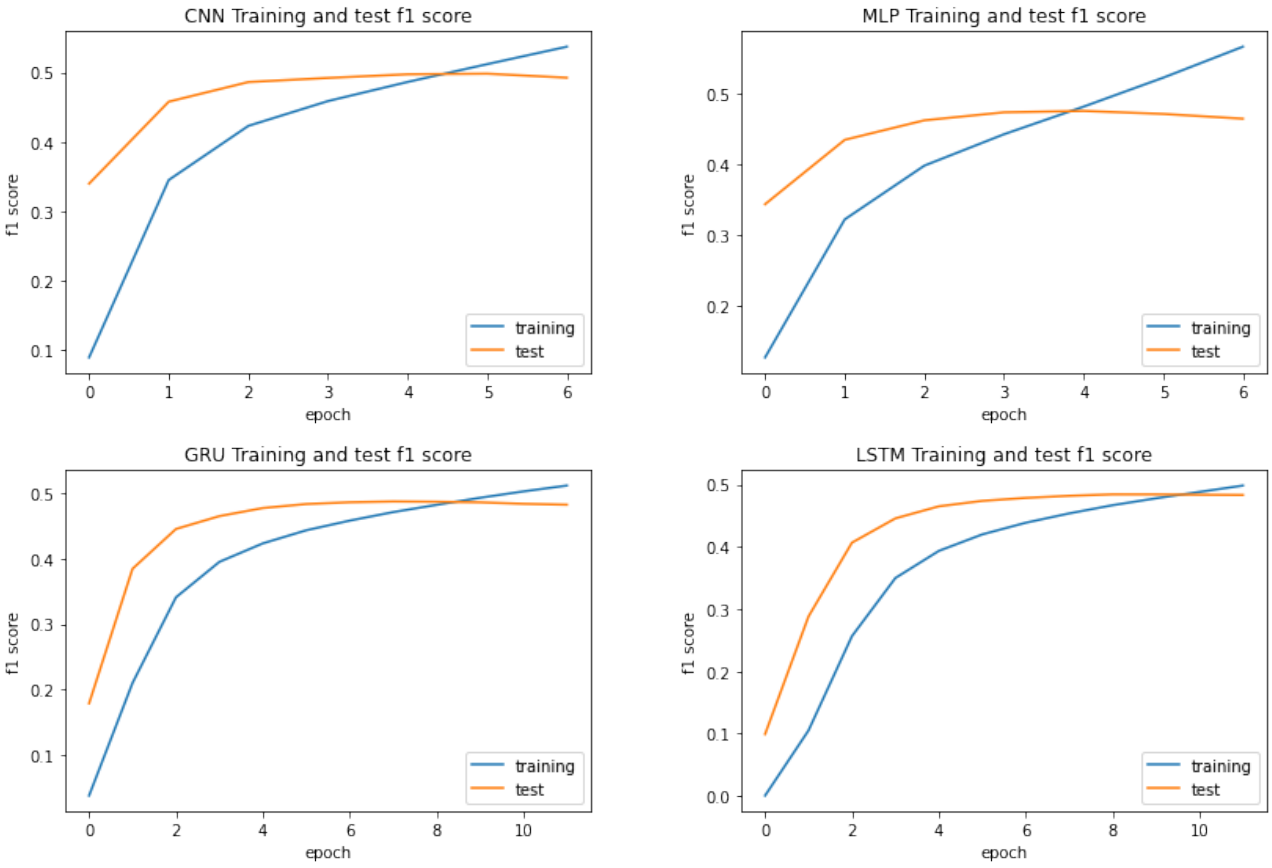

Figure 8.3: Training Test F1 score

8.4 Model F1 Result Comparison

The given plot 8.4 shows the f1 scores of various models based on epochs. At the beginning of the epochs, MLP performs the best, while LSTM is the worst. After some epochs, MLP lost its first position, and CNN stands at the top among all the models. GRU and LSTM perform well. MLP does not perform well after some epochs compare to others.

If we see the barplot 8.5, we see that CNN gives the best f1 score which is 0.4981. GRU takes second place with a 0.4842 f1 score. After that LSTM performs which is near 48.42. Though MLP starts with good results, finally MLP takes the last place with 0.4755 F1 scores.

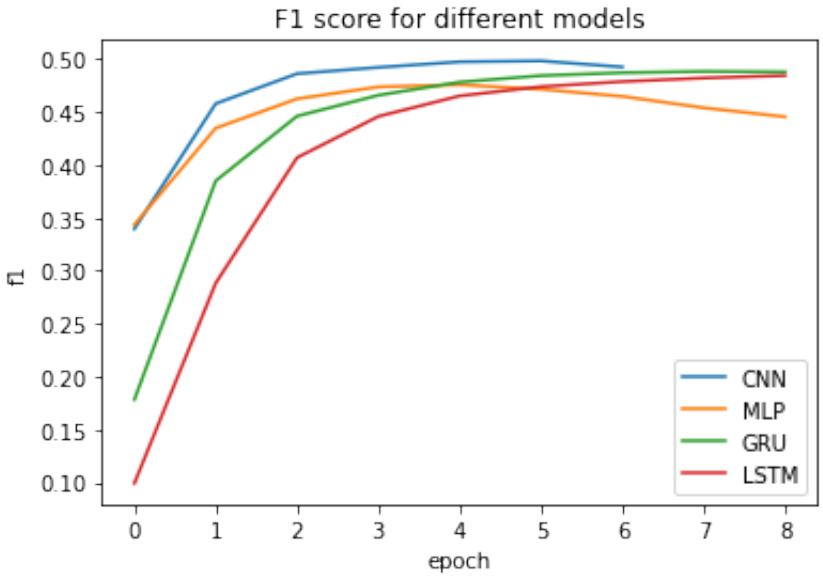

Figure 8.4: F1 scores result

Table 8.1: Result Comparison Table

Model	Full-Text F1	Title F1	Title F1(Experimental)
MLP	0.457	0.500	0.475
CNN	0.387	0.426	0.498
LSTM	0.363	0.466	0.482
GRU	NA	NA	0.488

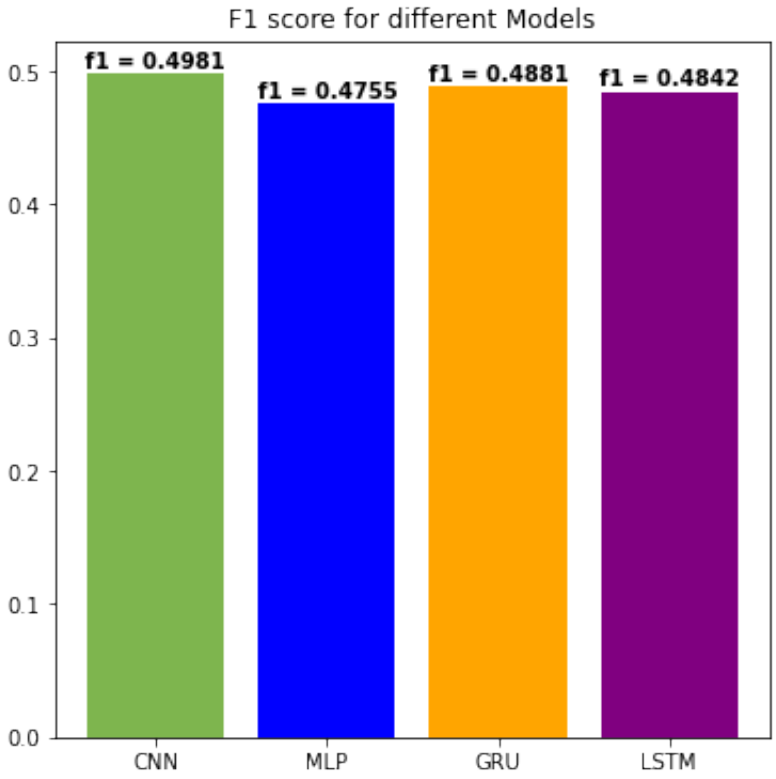

Figure 8.5: Model best F1 scores result

CHAPTER 9

Conclusions & Future Work

We have implemented 4 different types of the model including MLP, CNN, LSTM, and GRU. In the paper [12] only MLP performs better than us if we look at table 8.1. Without MLP, Our Three model performs best. In the MLP, they use the simple dense structure with 2k neurons. They get good results in the MLP, because of TF-IDF vector representation. This representation fails if the dataset becomes huge as it can not capture the semantic meaning. For title-based classification, semantic meaning is important. Our all model is trained with word embedding which is very popular method for text vectorization because it can capture the semantic meaning. The author uses the word embedding for CNN, and LSTM same as our model, but our model performs much better than theirs. Extreme Multi-label text classification is very difficult task in the natural language. It becomes hard because of the huge number of labels. So far, we did not achieve high benchmark like other classification task. We hope our contribution will help for future improving the model performance.

Bibliography

[1] R. Agrawal, A. Gupta, Y. Prabhu, and M. Varma. Multi-label learning with millions of labels: Recommending advertiser bid phrases for web pages. In *Proceedings of the 22nd international conference on World Wide Web*, pages 13–24, 2013.

[2] K. Balasubramanian and G. Lebanon. The landmark selection method for multiple output prediction. *arXiv preprint arXiv:1206.6479*, 2012.

[3] K. Bhatia, H. Jain, P. Kar, M. Varma, and P. Jain. Sparse local embeddings for extreme multi-label classification. *Advances in neural information processing systems*, 28, 2015.

[4] W. Bi and J. Kwok. Efficient multi-label classification with many labels. In *International conference on machine learning*, pages 405–413. PMLR, 2013.

[5] I. Chalkidis, M. Fergadiotis, P. Malakasiotis, N. Aletras, and I. Androutsopoulos. Extreme multi-label legal text classification: A case study in eu legislation. *arXiv preprint arXiv:1905.10892*, 2019.

[6] Y.-N. Chen and H.-T. Lin. Feature-aware label space dimension reduction for multi-label classification. *Advances in neural information processing systems*, 25, 2012.

[7] L. Galke, F. Mai, A. Schelten, D. Brunsch, and A. Scherp. Using titles vs. full-text as source for automated semantic document annotation. In *Proceedings of the Knowledge Capture Conference*, pages 1–4, 2017.

[8] F. Gargiulo, S. Silvestri, and M. Ciampi. Deep convolution neural network for extreme multi-label text classification. In *Healthinf*, pages 641–650, 2018.

[9] F. Gargiulo, S. Silvestri, M. Ciampi, and G. De Pietro. Deep neural network for hierarchical extreme multi-label text classification. *Applied Soft Computing*, 79:125–138, 2019.

[10] A. Joulin, E. Grave, P. Bojanowski, and T. Mikolov. Bag of tricks for efficient text classification. *arXiv preprint arXiv:1607.01759*, 2016.

[11] J. Liu, W.-C. Chang, Y. Wu, and Y. Yang. Deep learning for extreme multi-label text classification. In *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, pages 115–124, 2017.

[12] F. Mai, L. Galke, and A. Scherp. Using deep learning for title-based semantic subject indexing to reach competitive performance to full-text. In *Proceedings of the 18th ACM/IEEE on Joint Conference on Digital Libraries*, pages 169–178, 2018.

[13] R. Mihalcea, J. Chai, and A. Sarkar. Proceedings of the 2015 conference of the north american chapter of the association for computational linguistics: Human language technologies. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2015.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013.

[15] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272, 2014.

[16] Y. Prabhu and M. Varma. Fastxml: A fast, accurate and stable tree-classifier for extreme multi-label learning. In *Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 263–272, 2014.

[17] F. Tai and H.-T. Lin. Multilabel classification with principal label space transformation. *Neural Computation*, 24(9):2508–2542, 2012.

[18] I. E.-H. Yen, X. Huang, P. Ravikumar, K. Zhong, and I. Dhillon. Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification. In *International conference on machine learning*, pages 3069–3077. PMLR, 2016.

[19] K. Yoon. Convolutional neural networks for sentence classification [ol]. *arXiv Preprint*, 2014.

[20] R. You, Z. Zhang, Z. Wang, S. Dai, H. Mamitsuka, and S. Zhu. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. *Advances in Neural Information Processing Systems*, 32, 2019.