MULTI-LABEL CLASSIFICATION OF TEXT DOCUMENTS

USING DEEP LEARNING

HAMZA HARUNA MOHAMMED

SEPTEMBER, 2019

MULTI-LABEL CLASSIFICATION OF TEXT DOCUMENTS USING DEEP LEARNING

A THESIS SUBMITTED TO THE GRADUATE SCHOOL OF NATURAL AND APPLIED SCIENCES OF ÇANKAYA UNIVERSITY

BY:

Hamza Haruna MOHAMMED

Supervisor:

Asst. Prof. Dr. Abdül Kadir GÖRÜR

Co-Supervisor:

Asst. Prof Dr. Roya CHOUPANI

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE DEGREE OF MASTER OF SCIENCE IN COMPUTER ENGINEERING

SEPTEMBER, 2019

TITLE OF THE THESIS: **Multi-label Classification of Text Document using Deep Learning.**

SUBMITTED BY: **Hamza Haruna MOHAMMED**

Approval of the Graduate School of Natural and Applied Sciences, Çankaya University.

Prof. Dr. Can ÇOĞUN

(Director)

I certify that this thesis satisfies all the requirements as a thesis for the degree of Master of Science.

Prof. Dr. Sıtkı Kemal İDER

(Head of Department)

This is to certify that we have read this thesis and that in our opinion it is fully adequate, in scope and quality, as a thesis for the degree of Master of Science.

Asst. Prof Dr. Abdül Kadir GÖRÜR

(Supervisor)

Examination Date: 11.09.2019

Examination Committee Members:

Asst. Prof Dr. Abdül Kadir GÖRÜR **(Çankaya University)**

Assoc. Prof. Dr. Hadi Hakan MARAŞ **(Çankaya University)**

Assoc. Prof. Dr. İhsan Tolga Medeni **(Ankara Yıldırım Beyazıt University)**

STATEMENT OF NON-PLAGIARISM

I hereby declare that all information in this study has been obtained and bestowed in conformance with educational rules and ethics provided in educational medium. I can also state that, all materials and outcomes used in this thesis are referenced and cited in accordance with the educational rules.

Name, Last Name: Hamza Haruna MOHAMMED

Signature: 

Date: 11.09.2019

ABSTRACT

MULTI-LABEL CLASSIFICATION OF TEXT DOCUMENTS USING DEEP LEARNING.

MOHAMMED, Hamza Haruna

M.Sc., Computer Engineering Department

Supervisor: Asst. Prof. Dr. Abdül Kadir GÖRÜR

Co-Supervisor: Asst. Prof Dr. Roya CHOUPANI

SEPTEMBER 2019, 65 pages.

Recently, studies in the field of Natural Language Processing and some of its related important problem and Applications in the machine learning field continue to mount up. Machine Learning is prove to be predominantly data-driven in the sense that generic model buildings are used and then tailored to a specific application data. Needless to say, this has proven to be a very effective approach to modeling the complicated data dependencies we frequently experience in practice, making very few assumptions and allowing the information to talk for themselves. Examples can be found in chemical process engineering, climate science, systems, healthcare, and linguistic processing of natural language, to name a few. Moreover, text classification is one of the important aspect of Natural Language Processing. Text classification is the act of categorising text or text documents into a given set

of labels. While on the other hand, multi-label text classification deals with classifying text or documents into one more labels at the same time. Over the years, some methods for classifying text and documents have been proposed, including popularly known Bag of Words (BoW) method, Supervised Machine Learning, tree induction and label-vector embedding, to mention a few. These kind of tools can be used in many digital applications, such as document filtering, search engines, document management systems, etc. Lately, Deep Learning based methods is getting more attention, especially in an Extreme Multi-Label text classification. Deep learning is one of the major solutions to many machine learning applications that involve high-dimensional and unstructured data, such as pictures and text documents. However, it is of paramount importance in many of these applications to be able to reason accurately about the uncertainties associated with the predictions of these models. Therefore in this studies, we explore multi-label classification of text documents using deep learning methods such as CNN, RNN, LSTM, and even GRU. We investigate two scenarios in the studies. Firstly, multi-label classification models with plane embedding layer, and secondly with a Glove, Word2vec, and FastText as pre-trained embedding corpus for our models. We evaluate and compare these different neural network models performances in terms of multi-label evaluation metrics with respect to the two approaches.

Keywords: Natural Language Processing, Multi-Label Text Classification, Deep Learning, Word Embedding.

ÖZ

DERİN ÖĞRENME KULLANAN METİN BELGELERİNİN ÇOKLU ETİKET SINIFLANDIRILMASI.

MOHAMMED, Hamza Haruna
Yüksek Lisans, Bilgisayar Mühendisliği Bölümü
Tez Danışman: Dr. Öğr. Üyesi Abdül Kadir GÖRÜR
Ortak Danışman: Dr. Öğr. Üyesi Roya CHOUPANI

Eylül 2019, 65 sayfa.

Son zamanlarda, Doğal Dil İşleme alanında çalışmalar ve bununla ilgili bazı önemli problemler ve makine öğrenmesi alanındaki uygulamalar artmaya devam ediyor. Makina öğreniminin genel amaçlı modellerin uygulama alanina özel veri ile eğitilerek kullanılması ile veriye dayalı olduğu kanıtlanmıştır. Bu yöntemin pratikte sıkça karşılaştığımız karmaşık veri bağımlılıklarının modellenmesinde, çok az varsayımda bulunulduğunda ve bilgilerin kendileri için konuşması açısından çok etkili bir yaklaşım olduğu kanıtlanmıştır. Kimyasal proses mühendisliği, iklim bilimi, sistemler, sağlık hizmetleri ve doğal dilin dilbilimsel işlenmesinde bazılarına örnekler verilebilir. Ayrıca, metin sınıflandırma Doğal Dil İşlemenin önemli yönlerinden biridir. Metin sınıflandırma, metin veya metin belgelerini belirli bir etiket grubuna kategorize etme eylemidir. Öte yandan, çok etiketli metin sınıflandırma, metin veya

belgelerin aynı anda bir başka etikete sınıflandırılması ile ilgilidir. Yıllar içinde kelime çantası modelleri, denetimli makina öğrenmesi, ağaç azaltma ve etiket-vektör gömmeleri gibi metodlar önerilmiştir. Bu tür araçlar, belge filtreleme, arama motorları, doküman yönetim sistemleri gibi gerçek dünyadaki birçok uygulamada kullanılabilir. Son zamanlarda derin öğrenmeye dayalı modeller, bunların içinde de aşırı çoklu etiketli metin sınıflandırma modeli, ilgi çekmeye başlamıştır. Derin öğrenme, resim ve metin belgeleri gibi yüksek boyutlu ve yapılandırılmamış verileri içeren birçok makine öğrenimi uygulamasının ana çözümlerinden biridir. Bununla birlikte, bu uygulamaların birçoğunda, bu modellerin öngörüleriyle ilgili belirsizlikleri doğru bir şekilde aktarabilmek çok önemlidir. Bu sebeple, bu çalışmada çoklu etiketli metin sınıflandırma problemini evrişimsel sinir ağları, yinelemeli sinir ağları, uzun kısa zamanlı hafıza modelleri ve geçitli tekrarlayan birimler modelleriyle araştırdık. Bu çalışmada iki senaryo kulandık. Birincisi, gömme katmanıyla ve ikincisi Word2vec, Glove ve FastText gibi önceden eğitilmiş bir gömme bütüncesi ile çok etiketli sınıflandırma. Bu farklı sinir ağı modeli performanslarını, bu iki yaklaşıma göre çok etiketli değerlendirme ölçütleri açısından değerlendirdik ve karşılaştırdık..

Anahtar Kelimeler: Doğal Dil İşleme, Metin sınıflandırma, Derin Öğrenme, Makine Öğrenimi, Korpusu

ACKNOWLEDGEMENT

Firstly, I would like to express my sincere appreciation and gratitude to Prof. Dr. Erdoğan DOĞDU, Asst. Prof Dr. Abdül Kadir GÖRÜR and Asst. Prof Dr. Roya CHOUPANI as my supervisor and co-supervisors respectively, for their continuous guidance and counseling, supervision, suggestions and immense knowledge throughout the process of this masters study. Without your precious supports, it would not be possible to conduct this research, and make this thesis reach its conclusion.

My sincere acknowledgement also goes my thesis committee members Assoc. Prof. Dr. Hadi Hakan MARAŞ and Assoc. Prof. Dr. İhsan Tolga Medeni for their motivation as well as guidance towards the end of writing this thesis. Your insightful motivational comments encouraged me throughout this process.

Finally, I most express my profound sincere and gratitude to my family, especially my parents for their unparalleled love, encouragement, help and continuous support both financially and emotionally throughout my years of study and the process of this thesis. This attainment would not have been possible without them. I dedicate this accomplishment to them.

TABLE OF CONTENTS

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

NLP: Natural Language Processing

DL: Deep Learning

DNN: Deep Neural Network

ML: Machine Learning

NLU: Natural Language Understanding

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

Bil: Bidirectional

BoW: Bag-of-Words

CBOW: Continuous Bag-of-Words

FAIR: Facebook's AI Research

API: Application Programming Interface

S-VSM: Scrap Value Stream Mapping

HL: Hamming Loss

Acc: Accuracy

BR: Binary Relevance

CC: Classifier Chain

CSMLC: Cost-Sensitive Multi-label Classification

BPMLL: Backpropagation for multi-label learning

PD-Sparse: Primal and Dual Sparse

RakEL-d Random k-label

KNN: K-nearest neighbor

MLkNN: Multi-label k-nearest neighbor

MLkNN: Dependent Multi-label k-nearest neighbor

GPU: Graphics processing unit

TPU: Tensor processing unit

XMTC: Extreme Multi-label Classification

SLEEC: Sparse Local Embeddings for Extreme Multi-label Classification

C2AE: Canonical Correlated AutoEncoder

CHAPTER 1

INTRODUCTION

1.1 Motivations

With its contextual dissimilarity and unstructured form, the large quantity of textual information on the web and other platforms makes classification of text files of paramount significance. Nowadays, company decisions can be improved with the assistance of text classification through rapid decision-making and effective process automation. Extracting meaningful information from our unstructured texts in our email, social media, web pages, chats and many more is time consuming and hard at the same time. Although scientists have studied and even created a number of text classification applications, there is still a lack of a comprehensive overview of text classification studies in comprehensive format, particularly using latest trends in deep learning.

Deep learning is predominantly data-driven, in the sense that generic model structures are used, which are then adapted to the application-specific data. Needless to say, this has proven to be a very successful approach for modeling the complex data dependencies that we often encounter in practice very few assumptions are made and the data is allowed to speak for itself. However, another very common approach to modeling and prediction-making is to use simulation-based models based on physical insights or first principles. Examples are found in in healthcare, linguistic natural language processing (NLP), chemical process engineering, climate science, and autonomous systems, to mention a few. This approach to modeling

benefits from prior knowledge and can often leverage decades of research devoted to better understanding a specific problem area.

1.2 Natural Language Processing

Natural Language Processing (NLP) refers to the use of algorithms to extract semantic information from our natural language spoken speech or written text. It can be divided into two primary classes. The first one is Natural Language Generation(NLG). It can be describe as the concept of creating instruments for text or voice generation, while the second class is regarded as Natural Language Understanding (NLU). It can be define as the ability to understand sentences the spoken language buy an individual. NLP Applications can be found in Sentiment Analysis, Customer Service, Customer Service, Managing the Advertisement Funnel, and Market Intelligence etc.

1.3 Text Classification Problems and Applications

Text classification is a one of the primary research topic in Natural Language Processing (NLP). The concept of mapping a specified text or text document to a set of labels based on contextual categories led to the need to extract knowledge from a given text. All this due to the immense availability of textual data in our digital world, be it in industries, hospitals, and even over the web brought the attention of researchers and even software developers in the industries to texts related tasks, such as studies in the field of topic summarizing, clustering, categorization and classification of texts. These lead to many NLP application related the texts tasks like that of Search Engines, News Filtering, Spam Filtering, Document Organization and Retrieval System, Grammar Checker, and even Document Management Systems.

Moreover, most of the previous researches in text and documents classification used machine learning methodologies that has made a significant progress over the years. For instance partition method using tree induction together with label-vector

embedding in relation to the target space, Bag of Words (BoW), Supervised and Semi-supervised Machine Learning, and many other techniques available for text document classification [56]. However, challenges to those previous include handling those texts when the text is in large volume (Big Data), unstructured form of text data, and high complexity of our natural languages. With these challenges, it will be impractical to adopt classification on billions of text documents manually, which can result to intensive labor procedures and time consumption during the process [47].

Deep Learning-based semantic methods are getting more attention recently [48] [8] [21] [13]. The idea is to shift far from human-designed features of traditional machine learning method to an automatic information-context and sectional parts extraction of texts using neural networks based on deep learning models. However, most of the previous study approaches were base with multi-class classification binary and even binary classification. very little are base on an extreme multi-label classification problem techniques. In our study, we would explore the limitation of multi-label classification problem. Multi-label classification is the idea grouping an entity or sample to a given multiple label at a time. In other words, predicting toxic comment text documents that are not mutually exclusive. Therefore, we perform this multi-label classification experimental study base on distinct deep neural network classifiers and separate Natural Language methodologies like word embedding in order to observe different scenarios.

CHAPTER 2

TEXT CLASSIFICATION

Text Classification is one of the elementary tasks of Natural Language Processing (NLP) with a tremendous success in variety of applications, among others are intent detection, sentimental analysis, spam filtering and many more. It can be described as the act in which text is assigned or labelled with respect to a given tags or categories according to its content. In other words, it aims to assign predefined labels to text documents [101].

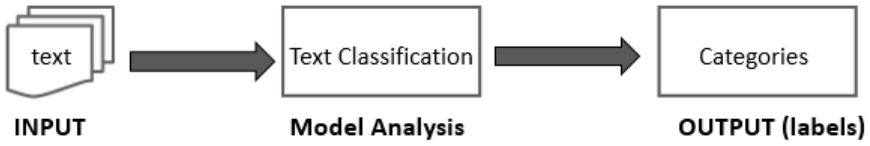

Figure 2.1: Text Classification Scenario.

As illustrated in Figure 2.1 text classification task can be described as follows. Let the training document be defined as:

$$D = d_1, d_2, ...d_n \tag{2.1}$$

such that the instances d_1 is assigned with a label l_1 from the set:

$$L = \{l_1, l_2, l_3, ..., l_k\}$$

The objection is to find a $classification$ model f where:

$$f : D \rightarrow L \qquad\qquad f(d) = l \tag{2.2}$$

Text Classification problem is divided into sub-different types, Binary Classification, Multi-class Classification, Multi-label Classification, and or Multi-label and Extreme Multi-label Classification. In this section, we will explore and elaborate on these types.

2.1 Binary Classification

Binary classification in other term binomial classification is the problem of assigning instances into a given label among only two label classes. The mapping scenario in which each corresponding label is associated with is described in equation 2.3 with respect to the basis of the classification rules.

Given a classification function:

$$f : D \rightarrow L \qquad\qquad f(d) = l$$

for which the label set L is :

$$L = \{1, 0\}$$

we can therefore map the following binary classification function as:

$$f(d) = \{0, 1\} \tag{2.3}$$

2.2 Multi-Class Classification

Multi-class classification is the task of classifying each text instance into one of three or more labels. The mathematical module for this type of classification problem can be describe as;

Given a classification function:

$$f : D \rightarrow L \qquad f(d) = l$$

for which the label set L is :

$$L = \{l_1, l_2, ..., l_k\}$$

we can map the following multi-class classification function for which instance d is labeled to a corresponding label among the given set of labels, as described below:

$$f(d) = \{1, 2, ..., K\} \tag{2.4}$$

2.3 Multi-Label Classification

Multi-label classification is more less the most complicated form of classification problem. One may find it difficult to differentiate it with multi-class. However, multi-label classification can be referred to as the a problem of assigning multiple labels to each given instance.

To describe this problem as a mathematical module as shown in Equation 2.5.

Let the given function f illustrated the mapping procedure for each instance to given set of labels:

$$f : D \rightarrow Z \qquad Z \subseteq L = \{l_1, l_2, ..., l_k\}$$

we can conclude that:

$$M = \{(x_i, Z_i)\}_{1,2\,,..,N} \tag{2.5}$$

$$x_i \in D \quad ; \quad Z_i \subseteq L$$

The main contrast between multi-class classification variants and the multi-label case is in the latter an instance can be assign between one or more of the labels at a given time among the class sets, while in the former an instance can be represented or assigned to more than label.

CHAPTER 3

MULTI-LABEL TEXT CLASSIFICATION TECHNIQUES

Several methods exits for multi-label classification. In this section, we broken down some of these methods used in multi-label classification. These methods include; Problem transformation methods, Leaning paradigms, supervised and semi-supervised methods, and using Adapted algorithms. We also highlight some of the literature of this methods, and give extensive overview of state-of-art tree based method, hierarchical, and deep learning methodologies.

3.1 Problem Transformation Method

Multi-label classification problems are challenging because of the high number of labels available, and the existence of label correlations problem in this type of problem. One way possible to handle this issue is to transform this given task into a traditional binary or multi-class classification problem. The idea here is to exploit the label correlations by simplifying the learning process. Some techniques under this Problem Transformation Method are introduced to address this issue, such as using Copy Transformation Method, Label Power Set Method, Binary Relevance Method, a and Ranking Pairwise Comparison Method. Now, lets clarify them one by one.

3.1.1 Copy Transformation

Copy Transformation strategy transforms and mapped the labels of multi-label instances into various single-label notations. For example, let the multi-label training example be described as:

$$(d_1, \{l_1, l_3\}), \quad (d_2, \{l_2, l_4\}) \quad and \quad (d_3, \{l_4\})$$

We can transform the problem into an binary classification of unweighted form $(d_i, l) \in D_k$ such that:

$$(d_{1a}, l_1), \quad (d_{1b}, l_3) \quad (d_{2a}, l_2), \quad (d_{2b}, l_4), \quad (d_{3a}, l_4)$$

Thus for all labels $\{l_1, l_2, ..., l_k\}$ training set instances $\{d_1, d_2, ..., d_k\}$ are transform to binary classification form. Based on that, a binary method or classifier can be applied to the training set such as in our case using Deep Learning method, or some other popular machine learning binary classification learning techniques like Random Forest [10] [12], Decision Trees, k-Nearest Neighbor [53] etc.

3.1.2 Label Power Set

Label Power Set method treats unique set of labels for a given instances in the multi-label problem as a single label or class. Pruning can be applied also here. To better illustrate this method, let us consider the previous multi-label example discussed in copy transformation method:

$$(d_1, \{l_1, l_3\}), \quad (d_2, \{l_2, l_4\}) \quad and \quad (d_3, \{l_4\})$$

The Label Power Set transform this labels set such that:

$$(d_1, l_{1,3}), \quad (d_2, l_{2,4}), \quad (d_3, l_4)$$

This method does not take into account the label correlations between the label classes, because it only consider each member of the labels power as a single label in the training set. And some other label combinations in the labels set may even have few positive samples. In this approach, the number of distinct label combinations grows exponentially as the number of class labels increase, thus make the method of high computational complexity.

3.1.3 Binary Relevance

Binary Relevance technique allocates a distinct classifier per different label in the labels set. In this method, each given classifier i use the entire dataset during training but only take note of a label of that given class i as a positive while the other label class as a negative. For example, given that:

$$(d_1, \{l_1, l_3\}), \quad (d_2, \{l_2, l_4\}) \quad and \quad (d_3, \{l_4\})$$

Binary relevance transform the original data q data sets D_{lj}, for $j = 1...q$ that contain all samples of the original data sets. If the label set of the original sample in the data set contain l_j, we labeled the new sample as positive, otherwise negative label will be annotated as indicated in the generated Table 3.1., Table 3.2, Table 3.3, and Table 3.4. After generating these set of data, we can apply classification on the instance, and the binary relevance outputs the union of the l_j labels that are positively predicted by q classifiers.

Table 3.1: Generated Binary Relevance (a)

Attributes	***Label***
d_1	l_1
d_2	$\neg l_1$
d_3	$\neg l_1$

Table 3.2: Generated Binary Relevance Data (b)

Attributes	***Label***
d_1	$\neg l_2$
d_2	l_2
d_3	$\neg l_2$

Table 3.3: Generated Binary Relevance Data (c)

Attributes	***Label***
d_1	l_3
d_2	$\neg l_3$
d_3	l_3

Table 3.4: Generated Binary Relevance Data (d)

Attributes	***Label***
d_1	$\neg l_4$
d_2	l_4
d_3	l_4

3.1.4 Ranking Pairwise Comparison

Ranking Pairwise Comparison learns a different classifier per each matching set of distinct labels. Using similar example described in Section 3.1.1 , let the multi-label training data be described as:

$$(d_1, \{l_1, l_3\}), \quad (d_2, \{l_2, l_4\}) \quad and \quad (d_3, \{l_4\})$$

Using ranking pairwise comparison method, the new dataset will be of binary label transformation form of $\frac{q(q-1)}{2}$, each pair of label is represented as (l_i, l_j), where $1 \leq i < j \leq q$. In each of the new data sets, the attributes or instances of the original dataset will be equally represented but annotated by at least one of the two labels only, and the binary classifier learns to distinguish the difference between the two labels and train each of these new data sets. At the last stage, each label is ranked with respect to the total votes received. Table 3.5 through Table 3.10 shows the data sets generated using ranking by pairwise comparison method.

Table 3.5: Generated Ranking Pairwise Comparison (a)

Attributes	***Label***
d_1	$l_{1,\neg 2}$
d_2	$l_{\neg 1,2}$

Table 3.6: Generated Ranking Pairwise Comparison (b)

Attributes	***Label***
d_1	$l_{\neg 1,3}$
d_3	$l_{\neg 1,3}$

Table 3.7: Generated Ranking Pairwise Comparison (c)

Attributes	***Label***
d_2	$l_{\neg 1,4}$
d_3	$l_{\neg 1,4}$

Table 3.8: Generated Ranking Pairwise Comparison (d)

Attributes	***Label***
d_1	$l_{\neg 2,3}$
d_2	$l_{2,\neg 3}$
d_3	$l_{\neg 2,3}$

Table 3.9: Generated Ranking Pairwise Comparison (e)

Attributes	***Label***
d_2	$l_{2,4}$
d_3	$l_{\neg 2,4}$

Table 3.10: Generated Ranking Pairwise Comparison (f)

Attributes	***Label***
d_1	$l_{3,\neg 4}$
d_2	$l_{\neg 3,4}$
d_3	$l_{3,4}$

3.2 Learning Paradigm

In Learning Paradigm methodology, almost all the existing multi-label classification methods can be classed on the basis of learning paradigms as batch learning and online machine learning. Batch learning schemes require all information samples being pre-accessible. It trains the model to use all the sample data and afterwards predicts the test sample using the found relationship. On the other hand, the online learning algorithms are increasingly building their models in sequential iterations. At iteration t, an online algorithm gets a sample x_t and predicts its label(s) in the present model; then the algorithm gets y_t, the true label(s) of x_t, and updates its model based on the instance's sample label pair and label.

3.3 Adapted Algorithms

Adapted Algorithms apply modification to the decision functions or cost functions and adapt a given single-label classification algorithms on the multi-label problem. Examples of these algorithms that adopt this scheme include, traditional K-nearest neighbor (KNN) algorithm [53], Decision Trees [29], BP-MLL adaptation method of back-propagation neural network algorithm [54] etc.

Additionally, [35] introduce a benchmark Multi-label Classification Algorithms. In their study, different multi-label classification algorithms and several benchmarks multi-label of different domain datasets with respect to the input space and label complexity. They performed hyper-parameter tuning in the experiment for both the problem transformation, example binary relevance (BR) and classifier chain (CC), random k-label (RakEL-d). Moreover, they use some adaptation method of multi-label classification algorithms like Multi-label k-nearest neighbor (MLkNN), Dependent MLkNN (DMLkNN), Backpropagation for multi-label learning (BPMLL) in the study.

3.4 Deep Learning Methods

Deep learning methods for multi-label and extreme multi-label start to gain popularity recently. One of the recently studied deep learning method is XML-CNN [26]. In their studies, they adopted the same strategy like in [21] [18] through adoption of different convolution filters in which the model pass through. They use dynamic max pooling scheme to capture more fine-grained features from different position or place in the document. Moreover, they opted for a binary cross-entropy as the activation function instead of the popular sigmoid output activation function adopted in extreme multi-label classification . Between the pooling and the output layers, they use extra hidden bottleneck layer between to capture the behavior of compact document representations, thus reducing the model size and boosting the model's efficiency. [55] introduced a different deep embedding scheme for extreme multi-label classification, and adopt non-linear embedding in both feature and label spaces. Their method model the feature space non-linearity and label graph structure simultaneously for the XMTC problem.

In addition to that, [19] developed Deep learning method and BoW method together. A representation of a document is built by averaging the embedding of the words appearing in the document, on which a softmax layer is applied to map the representation of the document to class labels. In building document depictions, it ignores word order and utilizes a linear softmax classifier. FastText is very effective to train, specially on multi-class classification benchmarks it achieves excellent performance. Nonetheless, the simple average architecture of word embedding input could curb its success in XTMC. This can be because document representation in XMTC needs to capture the much rich information in order to effectively predict various correlated labels and discriminate them from huge number of irrelevant labels. Also, another method for Multi-label Classification using deep learning state-of-art method is developed with a Rethinking Structure capability [46]. They proposed an algorithm in their model that imitate human thinking capability, to

handle label correlation and cost-sensitivity (that's cost importance of each distinct labels) by modifying the loss function in the model. The first step of the model is adopting Binary recurrence (BR) data transformation. They compare their model with other Cost-Sensitive Multi-label Classification (CSMLC) algorithms on different (domains) datasets. RethinkNet shows auspicious improvement compared to previous approach of multi-label classification using neuron networks (NN).

3.5 Other Methods

Furthermore, other methods use in this type of problem that does not fall into the earlier subsections include a recently proposed PD-Square [49] a dual sparse and primal method. In this study, the classification classifier penalise each label on the weighted embedding matrix. This makes the primal and the dual space to be very sparse, which is an advantageous to an Extreme Multi-label Text Classifications. The prediction time is linear for PD-Square, however the method come up with the solution to have sub-linear training time with respect to the number of labels using an algorithm called Fully-Corrective Block-Coordinate Frank-Wolfe training algorithm. PD-Sparse is claim to have less training time and model size compared to Logistic regression and Support Vector Machine on multi-label classification problems [49].

Additionally, SLEEC [5] presented a new embedding method that consists of 2 phases, learning embedding step and the k-nearest neighbor(KNN) classification step. The method sets training data into clusters, and the models learns embedding from each of the given clusters assembled. For that reason the search for each text document can be perform within the given cluster the document belongs to. High dimensional data clustering is prove to be rocky and unstable. Nonetheless, SLEEC introduced an ensemble scheme for clustering this dimensional data to improve the prediction accuracy.

Furthermore, [37] Tree-based method for extreme multi-label text classifications. The key idea in this method is to have comparable label distribution records in

each sub-set and characterize the distribution using a set-specific ranked label list. This is achieved by jointly maximizing the scores of the ranked label lists in the two sibling subsets for the Normalized Discounted Cumulative Gain (NDCG). In additon to that, they make multi-induced trees ensemble learning to enhance prediction robustness.

Moreover, Fusion based method introduced in [6] for Classification of text documents based on Pattern Recognition Letter along with score level fusion approach. They proposed and designed two different classifier approach using two different model representations for text documents. They use and enhanced Scrap value stream mapping (S-VSM) and interval-valued representation model for text documents representation. While Fusion two different classifier scores to haven effective text classification classifier that has an accurate result. They also use baseline neural network for unigram representation, specifically word level neural network for text representation for efficient capture of semantic information in text data. Other methods worth mentioning include; such as Semantic kernel Model, Concept Vector and Term based Vector Space Model, and other classification algorithms [52]. On top of that, Hierarchical attention networks for document classification method [47], Deep neural networks (DNN) based mode amp; Canonical Correlated AutoEncoder (C2AE) [48] other deep learning methods [8][13].

CHAPTER 4

METHODOLOGY

In the previous years, many different machine learning methods have achieved good accuracy in multiple research studies and projects related to classification problems, especially in binary and multi-classification cases. Recently, classification problem and its applications has attracted even more interest due to its applicability wide range of domains, especially multi-label text classification, scene and video classification, and bioinformatics [39].

Moreover, deep learning methods have been explored in many other machine learning solvable problems due to their high performance measure. With that, deep learning models are increasingly used in different machine learning domains including recently in text classifications [17]. However, multi-label classification is a bit complicated, comprising label correlation. Unlike the binary classification and traditional multi-class classification problem, multi-label classification deals with how to associate instance with a given subset of labels. And this type of classification problem can be tricky to handle especially with mere traditional simple approach use in binary and multi-class cases.

In this study, we used a famous python framework called keras to handled this multi-label classification task using supervised learning model - deep learning neural nets. The end-to-end classification pipeline methodology use in this study is described in Figure 4.1.

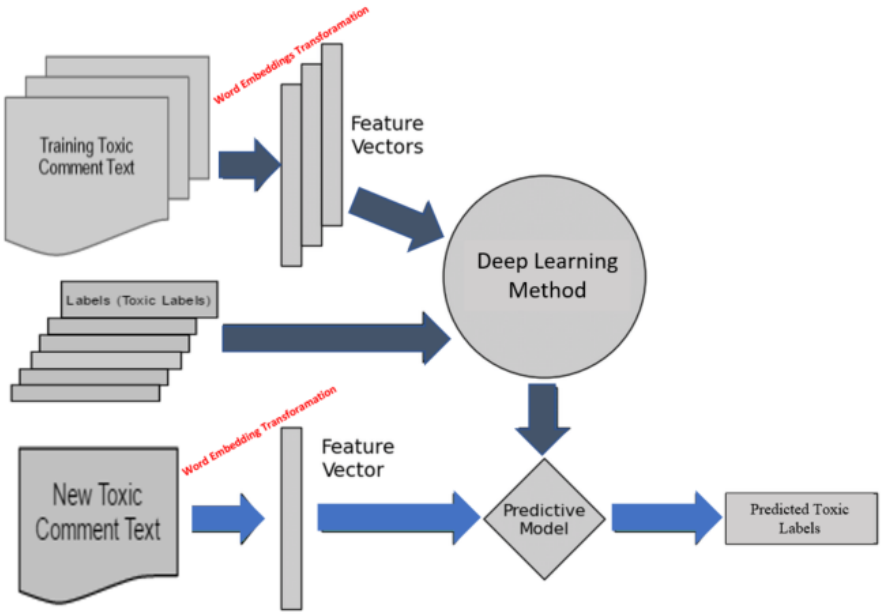

Figure 4.1: Deep Learning Supervised Leaning Model Methodology

- **Training Toxic Text Comment:** Input toxic text comment through which our models (supervised) are able to learn and predict desired labels.
- **Feature Vector:** Word embedding vector that contains the features of the input data. In our case; Word2Vec, fastText, and Glove.
- **Labels:** Predefined toxic labels/categories that our models will predict. Comprising of labels toxicity as described in Table 4.1.
- **Deep Learning Methods:** The algorithm-neural network that is used in the model for the text classification. In our study; we implement NN, CNN, RNN, LSTM, and GRU.
- **Predictive Model:** The trained model that is used on the test dataset to perform label predictions.
- **Predictive Toxic Labels:** Labels output from the model.

4.1 Deep Learning

Deep learning is a branch of machine learning that is getting lots of attention lately in both academia and the industries due to its high accuracy that were previously not achievable. Automatic feature extraction requires a structured or semi-structured large sets of labelled data and neural network architecture. Thus, deep learning models are train with large labeled set of labelled data employing deep neural network architectures that comprise of many different layers connected together. Deep Learning is synonymous to Neural Nets, inspired by the structure of cerebral cortext composed of various layers of interconnected perceptrons.

In a typical neural networks architecture [34] as illustrated in the Figure. 4.2 of four layer simple network, the figure shows a neural network consist of 4 layers. The first layer, as an input layer, and the two layers in the middle as the hidden layers, and while the last layer at the far end as the output layer with one output neuron (in our case with six toxic labels, most be six output neurons). In the network as describe in Figure. 4.2, the input values are feed to the network from the input layer, and then pass through the hidden layers until they converge to the last output layer called output neurons. The prediction of our output layer have 6 output nodes, given that the problem we are dealing with multi-label classification that have six (6) possible outcome of toxicity of comment in our studies. In the hidden layers, each node in the network has a weight that is multiplies with its given input value and keeps modifying the data over a few different layers until it feels what its relationship with the target output variable is ultimately.

Furthermore, the number of hidden layers comprises of 2-3 layers in a traditional networks, while in a deep networks can have more have more than 4 or up to 150 hidden layers. Thus, the amount of hidden layers available in neural networks usually refers to the term "*deep*". With deep learning, both feature extraction and modeling steps are automatic compared to machine learning technique of manually choosing

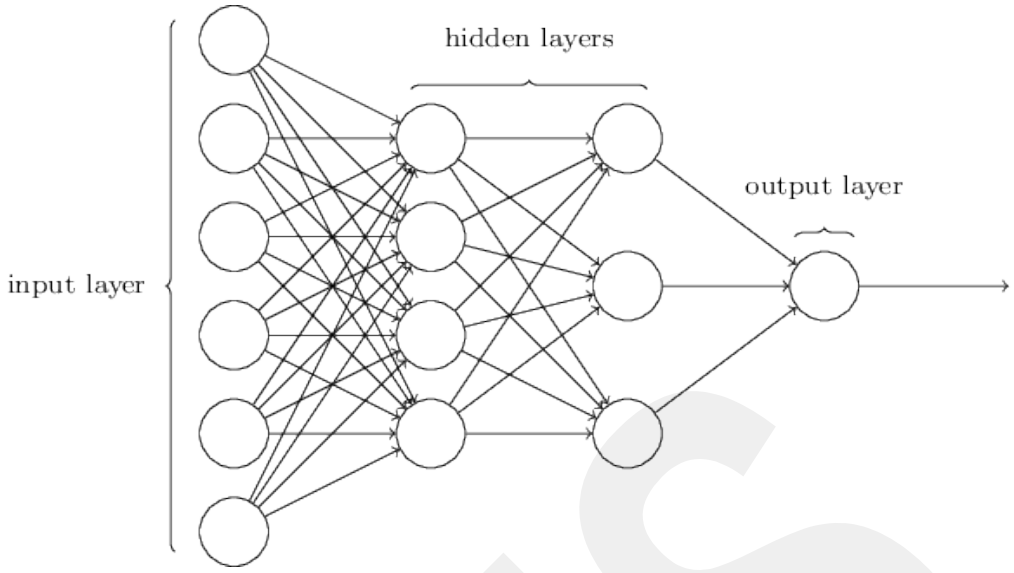

Figure 4.2: Neural Network Architecture.

the features and classifier. The key benefit of using deep learning is that as the size of the data increases, the model improves.

To begin with, we discussed the general multi-label classification problem in the section 2.3. The Wikipedia toxic comment dataset has 6 distinct labels, this means that given k samples:

$$D = \{d_1, d_2, ..., d_k\}$$

and labels

$$L = \{l_1, l_2, ..., l_k\}$$

with $l_i \in \{1, 2, 3, 4, 5, 6\}$. We then use different types of neural networks ranging from simple neural Network (NN), Convolutional Neural Network (CNN), Recurrent Neural Network(RNN), Long Short Term Memory (LSTM) network, Gated Recurrent Neural Network etc. as described in Section 4.1.1 till 4.1.5 sections to model the probability $P(c_j|d_i)$ of a class c_i with respect to d_i samples as show in Equation 4.1.

$$\widehat{l_i} = argmax_{j \in \{1,2,3,4,5,6\}} P(c_j|d_i) \tag{4.1}$$

The output layer of multi-class classification model is usually a softmax layer that

takes a vector z with K-dimension as an input vector of real values, and normalizes it into a K-dimensional $\sigma(\mathbf{z})$, that is proportional to the exponential in the range [0, 1] of real values, regarded as a probability values which will be sum up to be 1 [27]. The activation softmax function makes the neural network models to have multinomial distribution probability of a class c_j that is independent to other class probabilities. This method is good for binary or multi-class classification since they deal with single label prediction per sample.

Furthermore, suppose the model is to predict multiple label per sample, like in our multi-label classification problem. The softmax function results to fix number of labels prediction per sample. To handle this, we use a threshold value so that the probability between the labels be independent for a given samples. Sigmoid activation function as formulated in Equation 4.2 is the most generally used activation function for multi-label classification model. Therefore, we use the sigmoid function at the output layer of our models in order to eliminate this label dependencies between the classes.

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \tag{4.2}$$

for z $\in R$

Now, the probabilities of the neural network models of each class c_j is independent from the probabilities of the other class, and as a Bernoulli distribution as in shown in the Formula 4.3.

$$P(c_j|d_i) = \frac{1}{1 + \exp(-z_j)}. \tag{4.3}$$

In keras and tensorflow [16][1], the general loss function that is use to compile multi-class classification models is categorical_crossentropy loss [33], given the classification problem in thşs study has to do with more than two mutual exclusive targets, the targets should be encoded as one-hot vectors [38]. However, the target

classes can be of multiple form at once in multi-label classification problem. Therefore, we use binary_crossentropy in our models. We treat each output-neuron in the neural network as a separate random variable, and the product of the loss of those separate binary variables as the loss for the total or whole output vectors. In other words, we consider the product of binary cross-entropy for each single output unit as the loss for the labels.

4.1.1 Neural Network

In section 4.1, we cover all the basic details about neural network. Therefore, we implement a baseline neural network as illustrated in Figure 4.3 with six output neurons in the output layer to conform with our six multi-label toxic labels. The network takes the word-embedding vectors of the toxic comment as an input, and pass the hidden dense layers as described in the Table 4.2.

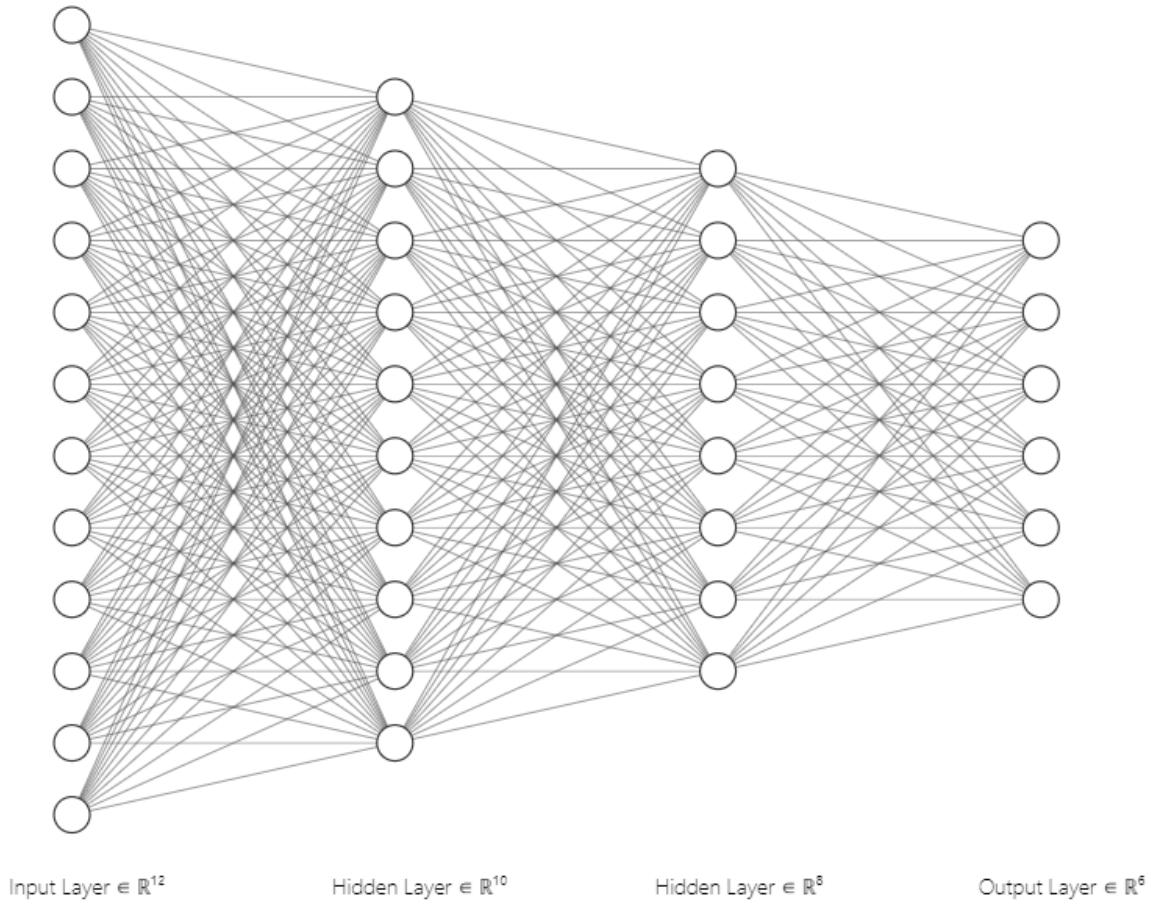

Figure 4.3: Neural Network Model Architecture.

4.1.2 Convolutional Neural Network

Convolutional Neural Network is one of the most widely used deep neural networks, especially in image related problems and applications. CNN uses 2D convolution layers and deform the input data with the learned features. This type of network architecture is well suited for processing 2D data, like image categorization, object classification tasks and other computer vision tasks [23]. In images problems, CNN automatically extracts features from pretrained features instantly. In this type of models and problem, as the network model trains, the network model also trains on the collection of those images. This ability makes this type of neural model popularly use in computer vision related tasks [43].

Moreover, Convolutional neural network starts to gain attention in other unstructured data tasks, such as text classification. Therefore, we build a CNN model for this experiment with model parameters described in Table 4.2 and the Figure 4.4 below.

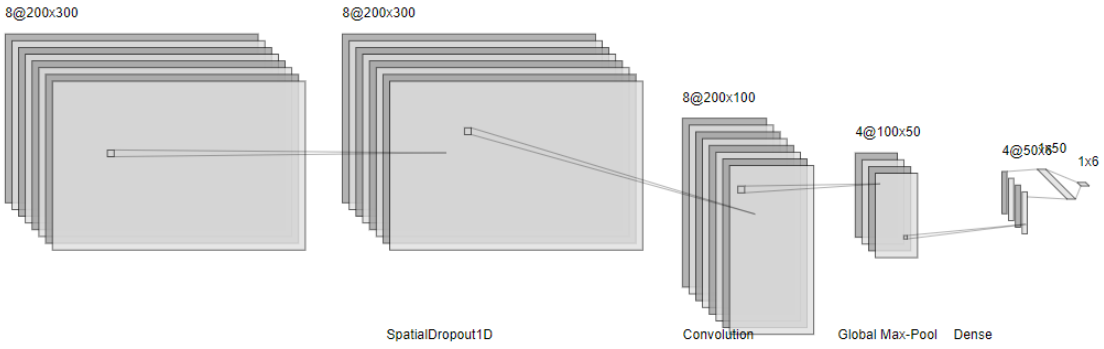

Figure 4.4: Convolutional Neural Network Model Architecture.

4.1.3 Recurrent Neural Network

Recurrent Neural Network (RNN) is a memory component structure network that distinguishes it from other general neural feed-forward networks. This distinction in the processing sequence of inputs makes it possible to compress prior inputs in a low-dimensional space, resulting in effective processing of text informa-

tion. RNN is that it can use information from previous time steps efficiently which is what edge it to other networks. That's why RNNs are applicable in the majority of NLP tasks. Figure 4.5 shows what a simple RNN looks like.

In the Figure 4.5, We've got our word vectors at the bottom (x_t, x_{t-1}, x_{t+1}). At the same time, each vector has a hidden state vector (h_t, h_{t-1}, h_{t+1}). In each RNN module, the hidden state is a feature of both the word vector and the prior step of the hidden state vector. This can be mathematically represented as in 4.4.

$$h_t = \sigma(W^{(hh)}h_{t1} + W^{(hx)}x_{[t]}) \tag{4.4}$$

As the Formula 4.4 indicated, the input input word vector x_t will be multiply by the weight matrix $W^{(hx)}$, and the hidden state vector at the previous time step will be multiply by the recurrent weight matrix $W^{(hh)}$. The recurrent weight is the same across the network, that same through all time steps. This is the one of the key difference to RNNs compared to other traditional NN. Each particular output of a given module $\hat{y}_t$ in the RNN, will be the product of h_t and W^S, which is another weight matrix as shown below in Figure. 4.5.

$$\hat{y}_t = softmax(W^{(S)}h_t) \tag{4.5}$$

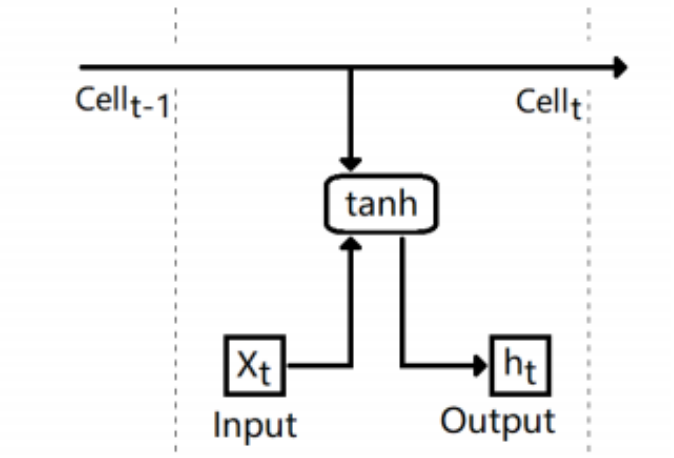

Figure 4.5: RNN Cell Structure.

Recurrent Network takes sequence of inputs (toxic words in our case) compared to traditional NN, say CNN that takes single input. However, the input word can be anywhere in the RNN type of deep network, ranging from a short toxic comment text to long toxic comment text. Moreover, the input order in this sequence can have a major impact during training on the state values of the vectors in hidden layer and the weight matrices. But the architecture makes the model to capture the information from past states during the training.

4.1.4 Gated Recurrent Unit

The first inception of Gated Recurrent Units is to provide way to eliminate the long term dependencies in tradition RNN. GRU come up with a complex solution to compute vectors of hidden state $h_{(t)}$ as described inSection 4.1.3. In RNN network architecture, error flows during backpropagation, from previous steps of time to present time step. Provide that the initial gradient value is a small-scale value (for example, say less than 0.25), the gradient will have virtually disappeared by the 3rd or 4th module and therefore the previous hidden states of the previous steps will not be updated.

besides

There are 3 component in GRU, the reset gate, update gate, and memory container. The reset and update gates are the functions of input values (toxix comment) besides hidden state values. This relationship can be mathematically describe as in be

The GRU provides a different way to compute the hidden state vector $h_{(t)}$ of our traditional RNN described in 4.1.3. GRU divide the computation into 3 component parts, an update gate, reset gate, and a new memory container. The first two gates (update ad reset gates) are both functions of the input toxic eord vector as well as the hidden state of the previous time step. This can be describe as Equation 4.6 to 4.8. The Equation 4.6 as the mathematical representation of update gate, while the Equation 4.7 as the rest gate formulation.

$$z_t = \sigma(W^{(z)}x_t + U^{(z)}h_{t-1}) \tag{4.6}$$

$$r_t = \sigma(W^{(r)}x_t + U^{(r)}h_{t-1}) \tag{4.7}$$

where;

z_t = Update gate.

r_t = Reset gate

The main difference here is that each gate used different weights. That's W^z and U^z for the update gate, and W^z and U^z for the other gate. The representation of the symbols of all the equation is describe in Table A in the Appendix A chapter. Furthermore, the Equation 4.8 describe tangent of the dot hadamard product of current update gate and previous state value of hidden vectors summed with input value at that step.

$$\hat{h_t} = tanh(Wx_t + r_t \; o \; Uh_{t1}) \tag{4.8}$$

With respect to the Equation 4.8, the dot indicates Hadamard product [28]. As the rest gate unit approaches zero, the whole terms becomes zero. Consequently, the information h_{t1} from the previous steps will be ignored. Then the unit will function with respect to the new toxic word vector x_t. Thus, the Equation of $h(t)$ is as follows:

$$h_t = z_t \; o \; h_{t-1} + (1 - z_t) \; o \; \hat{h_t} \tag{4.9}$$

h_t in Equation 4.9 is a feature of all three component parts: reset gate, update gate, memory container. The current hidden h_t state vector depends on the previous state vector z_t, when z_t approaches 1, the current memory container $(1 - z_t)$ will approach to 0. While as z_t is close to 0 or approaches 0, and we ignore the previous

hidden state, the current vector h_t state depends on the memory container at the present state.

4.1.5 Bidirectional Gated Recurrent Unit

This is a special network of Gated Recurrent Network. Bidirectional Gated Recurrent Unit allows the use of previous and future steps in order to predict the current state. It compose of only one input and forget gates, which makes it a one of the typical class of recurrent neural networks.

4.1.6 Long Short-Term Memory

Long Short-Term Memory are widely used in voice assistance and language transitional applications. Rampant reduction in RNN gradient value prevents weights from being updated by the Neural Network. However, this feature is enabled by the LSTM framework by avoiding a gradient issue. The LSTMs and GRUs have the special function of maintaining long-term dependencies in a sequence between words or phrases. These dependencies are recorded through gated that the sequence may ignore or store some information. The difference in number gates between GRU and LSTM is 2 gates for GRU, while there are 3 gates for the LSTM. This influences the amount of nonlinearities that the input passes through and ultimately impacts the general computation. The GRU also has not the same memory cell(c_t) as the LSTM. The graphical view of the LSTM Network is described as in Figure **??**.

The mathematical equation of LSMT can be illustrated as follows:

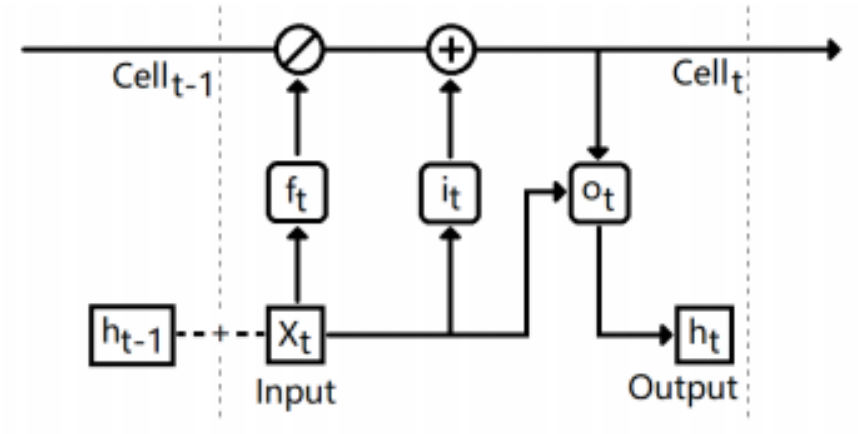

Figure 4.6: LSTM Cell Structure.

$$
\begin{aligned}
\sigma_t &= \sigma_g(W_\sigma x_t + U_\sigma h_{t-1} + b_\sigma) \\
i_t &= \sigma_g(W_i x_t + U_i h_{t-1} + b_i) \\
f_t &= \sigma_g(W_f x_t + U_f h_{t-1} + b_f) \\
C_t &= f_t * C_{t-1} + i_t * tanh(W_i x_t + U_i h_{t-1} + b_i) \\
h_t &= \sigma_t * \sigma_h(C_t)
\end{aligned}
\tag{4.10}
$$

Where;

x_t = input vector,

i_t = input gate.

f_t = forget gate.

σ_t = activation vector of output gate .

C_t = cell state.

h_t = LSTM cell unit output vector.

U and W = matrices.

The Equations 4.10 describe the mathematical model of Figure **??**. With reference to that same equation, forget gate f_i determine which cell state of C_{t-1} to get rid of, when the input x_t and h_{t-1} are pass to the cell of the Neural Network. After that, the values of x_t and h_t are then pass to i_t in order to save the updated values in the current state C_t. To minimise the loss function, the matrices W and U and b are periodically updated during the iteration. The output at each iteration and the

final one is computed using the activation vector of the output gate $sigma_t$.

4.1.7 Bidirectional Long Short-Term Memory

Bidirectional LSTM requires placing identical LSTM layers opposite to each other in the network as sketched in Figure 4.7. In this type of network, the input sequence normally allocate to the first layer and the other layer gets the reverse copy of the input sequence both in a bi-directional sequence. In other words, the input sequence are processed one by one, while the network steps through in both direction.

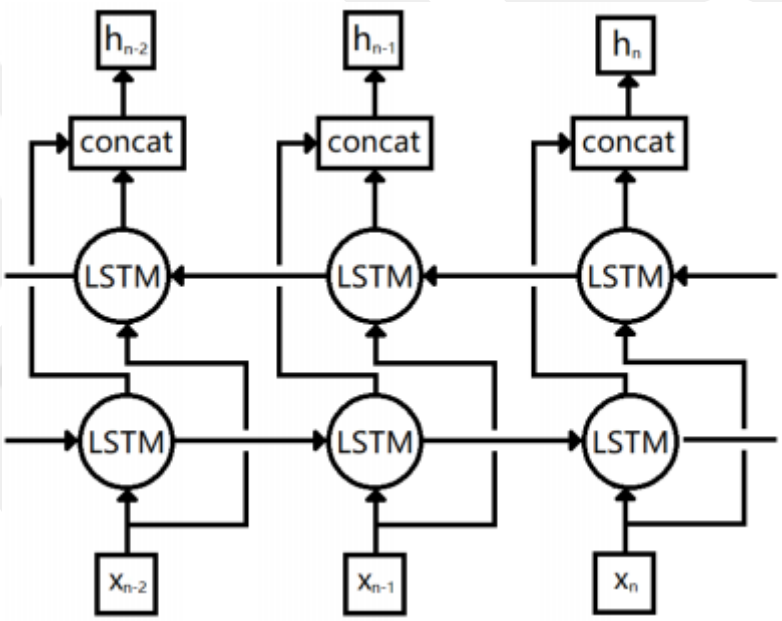

Figure 4.7: Bidirectional LSTM Structure.

The network ability to processed both previous and future states while training, results the network capable to interpret the context of the input easily. This approach was first introduced in voice and speech recognition, but it begin to instigate in natural language processing applications, especially in classification problems and predictive analysis of stock market [25]. Figures 4.5, 4.6, and 4.7 are sourced from [25].

4.2 Word Embedding Techniques

Extracting meaning from a word or text is one the primary aim of Natural language processing (NLP). Recently, words embedding methods of embedding words into a low-dimensional space were proposed using neural-network approach [4] [11]. Therefore, Word embedding is the essential part of any classification problem. This allows to represent the text-words using vector representation so that it can be fed to our neural networks. Dense vector representation is considered to be the state-of-art word embedding technique. The words in the text (toxic text comments in our case) are depicted by thick vectors where each vector represents the projection or mapping of the word into a constant vector space. Within the vector space, the position of each word is learned from the text within the surrounding words [32]. Bag-of-word model method was previously used in which a big sparse vector was used to represent each word in a vector form of a whole vocabulary.

The vector representation of the word or documents comprise of mostly zero values that make the vector sparse and of large space [36]. In our studies, we employed two (2) methods of embedding the toxic comment text into vectors. In the first method, we apply word embedding using keras embedding library [16] as the embedding layer in our models. This method is slower, and model dependent to our training toxic comment dataset. While the second method proposed in this study is to use pre-trained embedding corpus; Word2Vec, Glove and FastText in the experiments [51].

4.2.1 Keras Embedding

Keras machine learning library provides pre-built word embedding layers for neural network models. The input values in a typical Neural network model is an encoded integer values. Therefore, Keras provides Tokenizer API for this data preparations in order to feed the neural-network [16]. The Embedding layer is a flexible layer that learns to embed all specified words with random weights initialized

at first in the training dataset.

In addition, this embedding layer can be used to load a pre-trained word embedding model as a transfer learning form. It can also function together with the deep learning model itself to learn how to embed with the model. The term embedding can also be used as a learning paradigm that can be saved and used for the future in other models. Keras Embedding layer consists of 3 arguments in the architecture. These include input length, input dimension, output dimension. The input dimension specify the vocabulary size of the given text data, and output dimension argument stands for vector space size of the embedding words. And lastly, input length represent the input sequences length for the input layer of the Keras model. Over and above that, the Embedding layer has 2D vector output that can embed each as the input sequence of text or text documents [42].

4.2.2 Pre-Trained Embedding

4.2.2.1 Word2vec

Word2vec was first introduced by Google research team with idea of aggregating related models to produce word embedding [30]. The Word2vec algorithm they proposed [15] creates an Embedding vectors from words of text corpus that is more efficient compared to the previous approaches such as Latent Semantic Analysis approach of Count-based method [32]. In Count-base, the method compute statistical co-occurrence and frequency appearance of word with its neighbouring words in the text corpus, and map these count-statistics to dense vector for each given word found in the corpus. The associated models in Word2vec are two-layered, shallow neural networks designed to construct a linguistic representation of the words' contexts. It requires a big corpus of text as an input and output a vector-space of several hundred dimensions, with a corresponding vector allocated to each single phrase in the text corpus. In addition, words in the vector space that share common context in the corpus are placed close to each other [32].

Moreover, Word2vec can serve as a predictive model that can predict and learn

a word embedding from raw text. This model is computationally efficient similar to other Neural Probabilistic Language Models. The Word2vec algorithm comprise either two architecture for construction of dense embedding vectors of words representation, Skip-Gram model [24] and Continuous Bag-of-Words model (CBOW) [31]. These two models are close enough with each other in term of adopted methods and algorithms in both cases. he model predicts target word from source context words in continuous bag-of-words architecture. In this architecture, sequence of the context word does not effect the prediction of the word. While continuous Skip-gram architecture does the inverse, here prediction of source context words from a given targed word occur in this model. In this architecture, the distant source context-words are considered more that nearby source context-words. Therefore, CBOW performs faster than ht skip-gram method [31]. Visualization of skip-gram and CBOW models is describe in Figure 4.8 for better illustration.

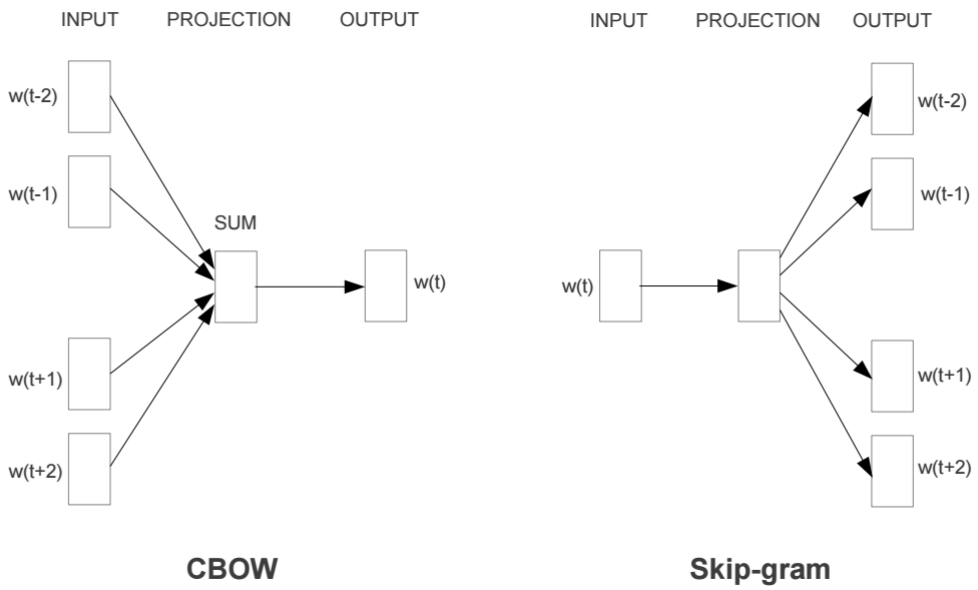

Figure 4.8: CBOW vs Skip-gram models. Source [32]

4.2.2.2 FastText

FasText created by Facebook's AI Research (FAIR) is an another pre-trained word embedding library for text classification. FastText model users neural network to create an supervised and unsupervised learning algorithms to build vector representations fro words [3].

Furthermore, fastText model supports training CBOW and Skip-gram models using softmax or hierarchical loss functions, and negative sampling. It perform very well for word representation of short sentence or text classification by modelling character level information for infrequent words in the text or sentence. In this model, apart from the original word, each word is depicted as a bag of character n-grams. This enables you to maintain or express denotation for prefixes/suffices and other short words that may appear for other phrases as ngrams [19]. In the course of model training specially during the model update, fastText model learns weights for the entire word token and each of the n-grams. In our experiment, we use fastText embedding library [20] trained on Wikepedia, Tatoeba, and SETimes datas sets as our pre-trained fastText.

4.2.2.3 Glove

Global Vectors for Word Representation (Glove) is a model for distributed word representation that is first developed by Stanford. Glove is an unsupervised model for acquiring semantic vector representations of words. The method combine both local context window word embedding methods together with global matrix factorization method to obtain word embedding in a linear substructures form. In this method, a corpus build a word to word co-matrix and aggregate the representation as the word embedding [36]. The difference between Glove and Word2ec is that, the later rely not only local context information of words statistics, but it acquire word vectors from the combination of statistics for global word co-occurrence.

In Glove, log-bilinear model takes into account the probabilities of word to word co-occurrence ratios with likelihood or possibility to encode some semantic meaning

from the given words relation. In addition to that, the train objective in this model is to have a trained vectors of words, given that the value of the dot product between given words is equal to logarithmic value of co-occurrence of of words likelihood. The Glove word vector representation performs better than in some Word2vec analogy problem

4.3 Dataset

Toxic Comment Classification dataset consists of Wikipedia Comments from Wikipedia Talk Pages which Conversation AI team asked 5000 crowd-workers labeled the comments according to their relative toxicity comments labels such as "*toxic*", "*severe_toxic*", "*obscene*", "*threat*", "*insult*", "*identity_hate*" [44]. This data set is multi-label text related problem type, and it consists of approximately ∼160k observation in total, ∼125k with zero labels (toxicity) of any type, and approximately ∼35k classified in one or more toxicity categories. The dataset characteristics is described in Table 4.1 and the following data features:

- Number of total data points **159571.**
- Observations in one or more class **35098.**
- Unclassified observation **124473.**

Furthermore, the format and the descriptions of the 3 data-files is explain in appendix B, reference to Tables B.1, B.2, and B.3. The Table B.1 describe the train data, while Table B.2 and B.3 shows the two test data files for this data set.

4.3.1 Data Preparation and Pre-processing

The dataset is an unbalanced-dataset, and the existence of high occurrence of 124473 unclassified observation. This limitation could affect model. Therefore, we used the 16225 samples that are classified in at least one sample of around 35098

Table 4.1: Dataset Characteristics

Label Type	***Data points***
toxic	15294
severe_toxic	1595
obscene	8449
threat	478
insult	7877
identity_hate	1405

in order to train and validate our model. We then removed unwanted punctuations and characters in both train and test data, so that would not affected our models.

4.3.2 Experiment Data Splits

That splits ratio for train, validation, and test data on most of the machine learning experimental data relies on the number of samples in the dataset, together with the type of model selected for the experiment. It's always good to split the data into $3:1:1$ ratio if the experimental dataset is 1 file. But thanks Kaggle [50] idea of separating train and test data into separate dataset files, we only needed to splits the train dataset into train and validation splits data, and then treat the test dataset as our test split data. In our experiment, we used cross validation [40] to split the train data into random train and validation subset, specifically Keras k-fold cross validation method [45]. Finally, the models were trained and validated iteratively on these different sets of data.

4.4 Model

This section elaborate on the structure of our deep neural network models selected for the experiment. We also highlight the functionalities and parameters use in training a neural network, such as activation function and optimizer.

4.4.1 Models Parameters Summary

While training the network, we apply a callback function that allows us to specify the performance measure to monitor, using a specific number of training epochs. The model stop the training process, when the model performance stops improving on a hold the validation dataset.

Table 4.2: Classifier Special Filters And Dense Layers for the the Models.

Model	***Special Filters***	***Dense Layers***
NN	NA	50, 6
CNN	#filters = 100, kernel size = 4 ×100	50, 6
RNN	Units = 25	50, 6
LSTM	Units = 25	50, 6
Bidirectional LSTM	LSTM units = 25	50, 6
GRU	Units = 128, filters = 64, kernel size = 4 ×64	6

Bidirectional GRU	#GRU Units = 128,
	filters = 64,
	kernel size = 4 ×64
		6

4.4.2 Activation Functions: ReLU and Sigmoid

In deep neural network terms, Activation functions are used to convert weighted input sums into an output value in a fully connected layer.

The first activation function used in this studies is ReLU activation function as it is mathematically define as in the Equation 4.11.

$$ReLU(x) = max(0, x) \tag{4.11}$$

The function helps counteract gradient disappearing issues often associated with back-propagation in profound and complicated neural networks. Therefore, it provides more effective way of computations when training.

Sigmoid activation function 4.12 is the other activation function we use in the neural networks. The activation function (Sigmoid) has a "s"-shaped curve and a variety rage of [0, 1]. The function can be mathematically represented as showned in Equation 4.2 and the Formula 4.12 below.

$$\sigma(z) = \frac{1}{1 + \exp(-z)} \qquad for \quad z \in R \tag{4.12}$$

With respect to the function 4.12, when x is approaches to zero, the function behave in a non-linear, smooth function with steep slope. Which implies that even modifications in predictors are small, it can demonstrate clear differences of response values. Because we are dealing with binary classification, it's crucial for the activation function to side with either 0 or 1.

4.4.3 Adam Optimizer

Although SGD is faster in helping descent of gradient, its precision and accuracy is not as desirable as it should be. In our experiment, we introduce different optimizer called Adaptive Moment Estimation (Adam) [22]. The optimizer concept of RMSprop and Momentum. Momentum utilizes an exponentially weighted gradient average to fix the oscillation issue of gradient descent. Sometimes, gradients take too many downward oscillations to achieve the local minimum, because we calculate in mini-batches when calculating gradients in SGD, which does not always lead immediately to the local minimum. And increasing the learning rate of the gradient descent can cause the gradient to overrun and even diverge in its descent.

On the other hand, RMSprop splits the learning rate by an exponentially declining square gradient average that dramatically decreases the learning rate as gradients approach the minimum. For the gradient descent, Adam calculates an

adaptive learning rate. It calculates not only an exponentially declining average of square gradients such as RMSprop, but also an exponentially declining average of gradients such as Momentum [41].

4.5 Evaluation Metrics

Metrics measures of multi-label classification differ to that of single-labeled binary and multi-class classification evaluation measures. In the latter case, simple measures like accuracy, precision, recall are been used. However, in multi-label actual labels subset prediction is more considered than no prediction occurrences.

To evaluate a multi-label model or learning given document data set denoted as M:

$$M = \{(x_i, Y_i)\} \qquad x_i \in D; Y_i \subseteq L$$

and the *labels*

$$L = \{\lambda_j : j = 1, ..., q\}$$

the prediction denoted by Z for instances in M be

$$Z_i \subseteq L$$

We then applied the following metrics for the multi-label classification performance for our experiments.

4.5.1 Hamming loss

Hamming loss evaluation metrics measure the proportion of labels associated with an unpredicted instance or sample. That is the rates at which the labels are misclassified [9].

Thus, the formula below illustrates the mathematical formation of Hamming loss as a fraction of the incorrect labels for the overall number of labels predicted:

$$HL = \frac{1}{N}\sum_{i=1}^{n}\frac{\mid Y_i \mid + \mid Z_i \mid -2 \mid Y_i \cap Z_i \mid}{\mid L \mid}$$

Therefore, we can conclude that;

$$HL = \frac{1}{N}\sum_{i=1}^{n}\frac{\mid Y_i \triangle Z_i \mid}{\mid L \mid} \tag{4.13}$$

The optimal value of this evaluation is zero, given that we are dealing with a loss function as its shown in the Equation4.13.

4.5.2 Accuracy

Accuracy measurement calculates among the entire expected labels the accurately predicted true labels. For most classification problems, This is more balanced and better performance measure compared to the hamming loss [14]. The Equation 4.14 describe the mathematical formula for this evaluation measure.

$$Acc = \frac{1}{N}\sum_{i=1}^{n} I(Z_i = Y_i) \tag{4.14}$$

$$For;$$

$$I(true) = 1; \quad I(false) = 0$$

4.5.3 Precision

Precision calculates the percentage of classification out of all positive classification as described in the mathematical formulation below 4.15.

$$P = \frac{1}{N}\sum_{i=1}^{n}\frac{\mid Y_i \cap Z_i \mid}{\mid Y_i \mid} \tag{4.15}$$

4.5.4 Recall

Recall shows the classifier's ability to classify the results as positive when the subjects are genuinely positive. Recall in classification is also referred to as *sensitivity*. This can be calculated using the Equation 4.16.

$$R = \frac{1}{N} \sum_{i=1}^{n} \frac{\mid Y_i \cap Z_i \mid}{\mid Y_i \cup Z_i \mid} \tag{4.16}$$

4.5.5 F1-measure

This measure can be describe as generalized macro F1-score form that can be computed using the Equation 4.17. It is expressed as the weighted harmonic mean measures of recall and precision, where a measure of F1 reaches its greatest value at 1 and the worst score at 0. In multi-label case, this can be calculated as the average of the f1-measure of each label or class with weighting with respect to the average parameter.

$$F_1 = \frac{1}{N} \sum_{i=1}^{n} \frac{2 \mid Y_i \cap Z_i \mid}{\mid Z_i \mid + \mid Y_i \mid} \tag{4.17}$$

CHAPTER 5

EXPERIMENTS

This section describe the necessary configuration setups and the result for this thesis study experiments.

5.1 Experimental Setup

In our experiment, we use Google-Colab Cloud Service that supports Keras and TensorFlow [7]. On top of that, the notebook support fast and high acceleration tools like GPU and TPU. Therefore, we configured our experiment to run on Python-3 programming language as our run-time type and GPU hardware accelerator tool for our notebooks in the experiment as shown in Table 5.1 and the notedbook [1]. The code and experiment of this studies can be found in GitHub repository [2].

Table 5.1: Experiments System Specification Setup in Colab.

	SPECIFICATIONS
GPU	1xTesla K80 , compute 3.7, having 2496 CUDA cores , 12GB GDDR5 VRAM
CPU	1xsingle core hyper threaded Xeon Processors @2.3Ghz i.e(1 core, 2 threads)
RAM	12.6 GB Available
Disk	33 GB Available

[1]https://colab.research.google.com/drive/151805XTDg–dgHb3-AXJCpnWaqRhop$_2$
[2]https://github.com/Hamxea/Multi-label-Classification

5.2 Experimental Results

In this experimental studies, we evaluate our models with respect to F1-measure, Precision, Recall and AUC evaluation metrics described in Section 4.5. We also compare and contrast our findings with [2] to see the behavior of measures in terms of that previous study. The F1-measure or F1 score is the harmonic mean of precision and recall. Our models obtain an ideal score by merely assigning each class to each input by using accuracy (precision) only. Therefore, a metric should also penalize inaccurate class predictions (recall) to prevent this. We use F-beta score ranging from 0 to 1 as a weighted mean of the true class predictions versus the ratio of the false class predictions. As shown in Table 5.2 our model precision and F1-measure outperforms the compared study by more than 8% in both Bidirectional LSTM and GRU models using Glove and FastText pre-trained embedding, in the other models contrast with at least 5% . This shows that the percentage of how many selected labels are more relevant in our models than compared study. With reference the table, their model has a greater recall rate with distinct classes across the board, which implies that it is considerably easier to predict toxic. In general, however, it has lower precision, which means it is underfitting in classification of clean comments.

Furthermore, our results indicate that the importance of assigning correct classes when beta less than 1 for the models based on table. And when beta is greater than 1, the F1-measure shows that the models are instead weighted to penalize wrong class predictions. The result also shows that the pre-trained embedding does not apply significant increase to all the evaluation measures in the models. Only Global Vectors for Word Representation (Glove) maintain a uniform increase in contrast to the models without pre-trained embedding and the other two pre-trained embeddings (Word2vec and FastText). Moreover, AUC achieve highest score in using Glove in all the models experimented with respect to Table 5.2.

Overall, Long Short-Term Memory (LSTM) outperforms almost all the others

Table 5.2: Experimental Results and Comparison with Previous Work [2]
P: *Precision*, **R:** *Recall*, **F1:** *F1-Measure*, **AUC:** *Area Under Curve*

	Our Models Metric Results				**Metrics Comparison**			
	P	**R**	**F1**	**AUC**	**P**	**R**	**F1**	**AUC**
NN	.88	.84	.858	.921	-	-	-	-
CNN	.86	.83	.842	.907	-	-	-	-
CNN ($fastText$)	**.86**	.83	**.778**	.904	.73	**.86**	.776	**.981**
CNN ($Glove$)	**.85**	.76	**.795**	.912	.70	**.85**	.748	**.979**
CNN ($Word2vec$)	.82	.73	.739	.889	-	-	-	-
RNN	.87	.80	.829	.913	-	-	-	-
RNN ($fastText$)	.76	.70	.729	.875	-	-	-	-
RNN ($Glove$)	.83	.72	.776	.923	-	-	-	-
RNN ($Word2vec$)	.77	.73	.719	.882	-	-	-	-
LSTM	.88	.85	.862	.932	-	-	-	-
LSTM ($fastText$)	**.87**	.75	**.796**	.917	.71	**85**	.752	**.978**
LSTM ($Glove$)	**.86**	.76	**.803**	.930	.74	**84**	.777	**.980**
LSTM ($Word2vec$)	.84	.73	.744	.896	-	-	-	-
Bil LSTM	.90	.87	.883	.943	-	-	-	-
Bil LSTM ($fastText$)	**.87**	.76	**.804**	.919	.71	**.86**	.755	**.979**
Bil LSTM ($Glove$)	**.87**	.79	**.843**	.942	.74	**.84**	.765	**.981**
Bil LSTM ($Word2vec$)	.84	.74	.753	.899	-	-	-	-
GRU	.81	.71	.753	.885	-	-	-	-
GRU ($fastText$)	.88	.80	.844	.934	-	-	-	-
GRU ($Glove$)	.86	.82	.843	.942	-	-	-	-
GRU ($Word2vec$)	.86	.81	.795	.916	-	-	-	-
Bil GRU	.83	.73	.776	.898	-	-	-	-
Bil GRU ($fastText$)	**.88**	.82	**.845**	.937	.72	**.86**	.765	**.981**
Bil GRU ($Glove$)	**.87**	.83	**.852**	.947	.73	**.85**	.772	**.981**
Bil GRU ($Word2vec$)	.87	.83	.807	.922	-	-	-	-

models in the four evaluation measure metrics, then followed by CNN, Bidirectional GRU (with FastText and Glove) and baseline Neural Network (NN) models. However the scores fluctuate as is shown in the Figure Table 5.2, due to the imbalanced identity of the Toxic Comment data set. Nevertheless, this results outperform the previous researches on this data in terms of precision and F1-measure.

We can also infer from the Accuracy and Loss plots diagrams for training and validation measures as shown from Figure. 5.1 to Figure. 5.7 that without the callback function implemented and used in these experiments, rigorous overfitting and underfitting would have exits in some of the models. This can be due to the imbalanced nature of the data set. For example, in Figure. 5.1 to Figure. 5.4 the training accuracy increases as the validation accuracy steadily decreases with negative slope. These show that overfitting is likely to occur in these models, therefore it is good thing that we use the early stopping or callback function described in Section 4.4.1. While in the other plots, the validation accuracy plot in the diagram shows the existence of little over or absence of overfitting in the neural network models, like in Figure. 5.7 to Figure. 5.25, the plot shows increase in validation accuracy.

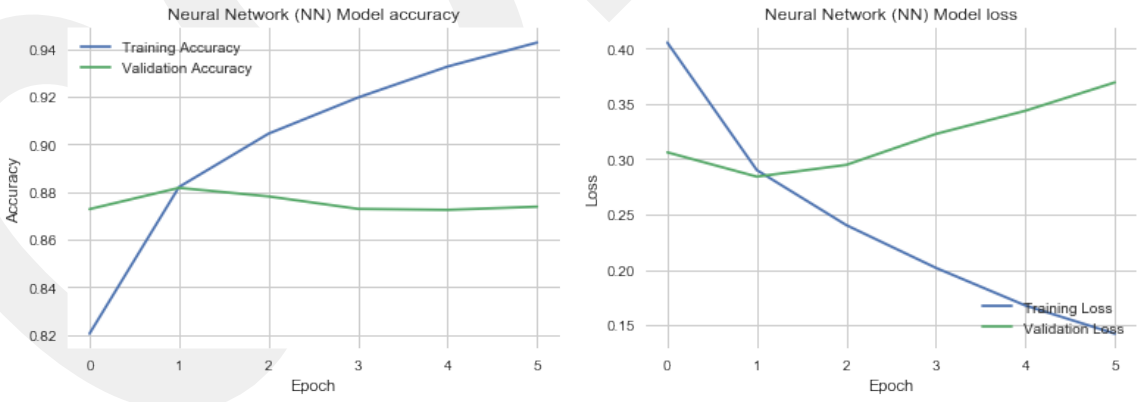

Figure 5.1: Baseline NN Accuracy and Loss Plots

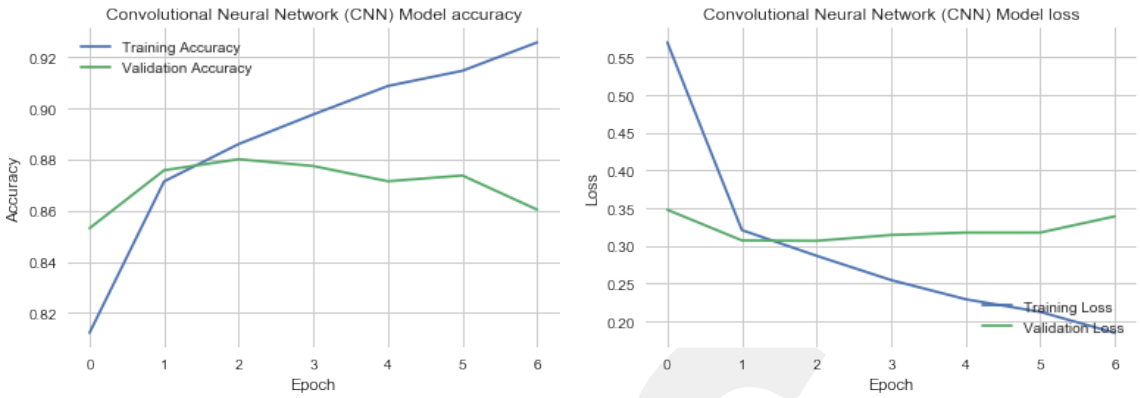

Figure 5.2: CNN Accuracy and Loss Plots

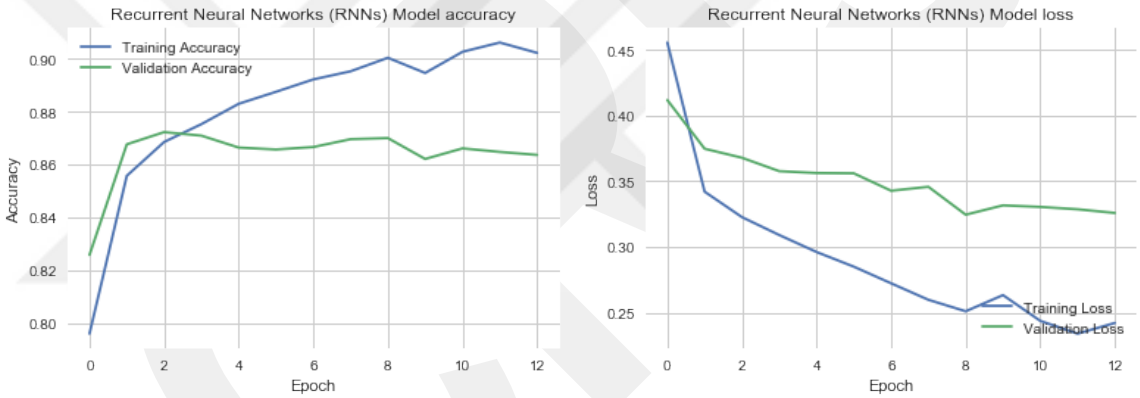

Figure 5.3: RNN Accuracy and Loss Plots

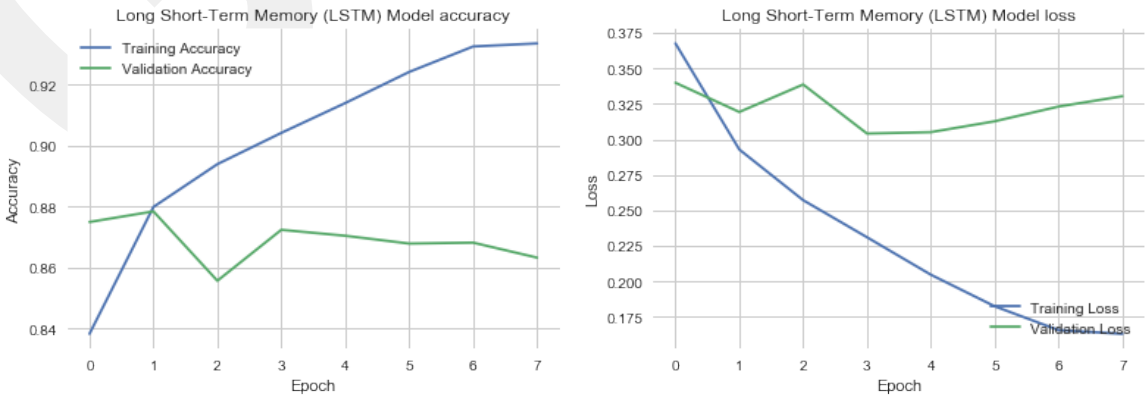

Figure 5.4: LSTM Accuracy and Loss Plots

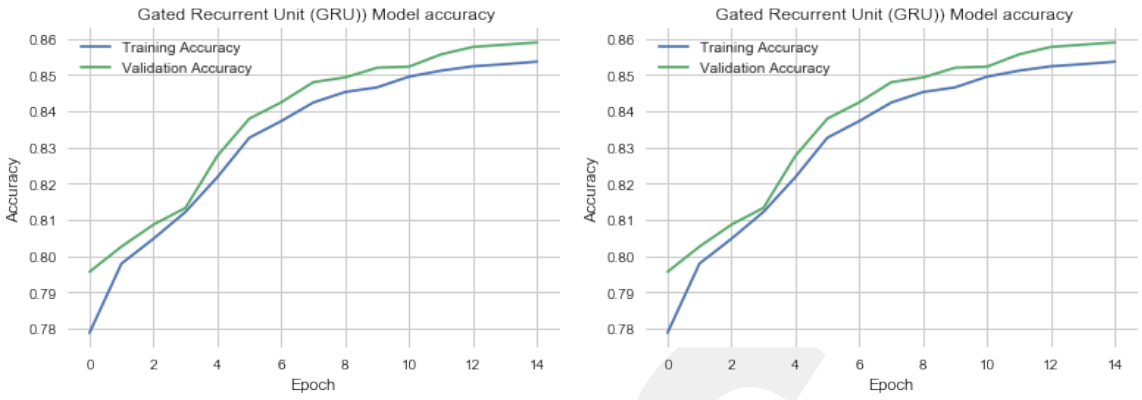

Figure 5.5: GRU Accuracy and Loss Plots

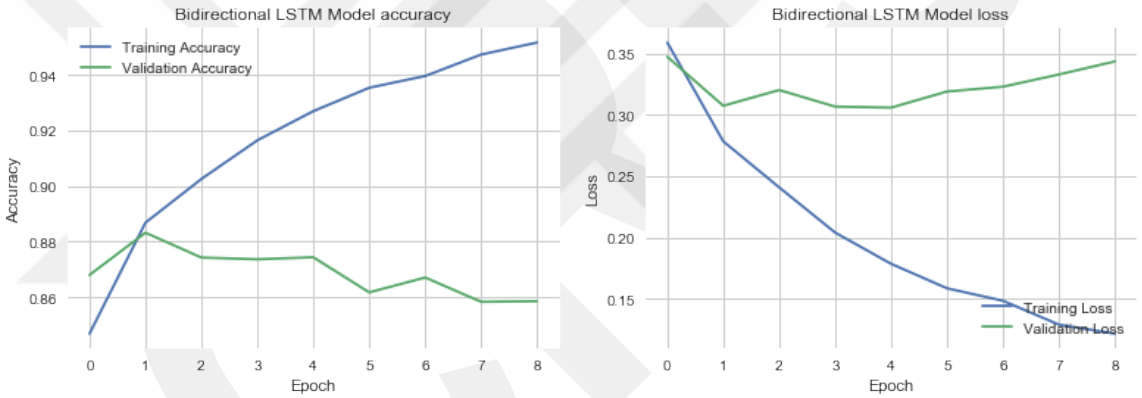

Figure 5.6: Bil-LSTM Accuracy and Loss Plots

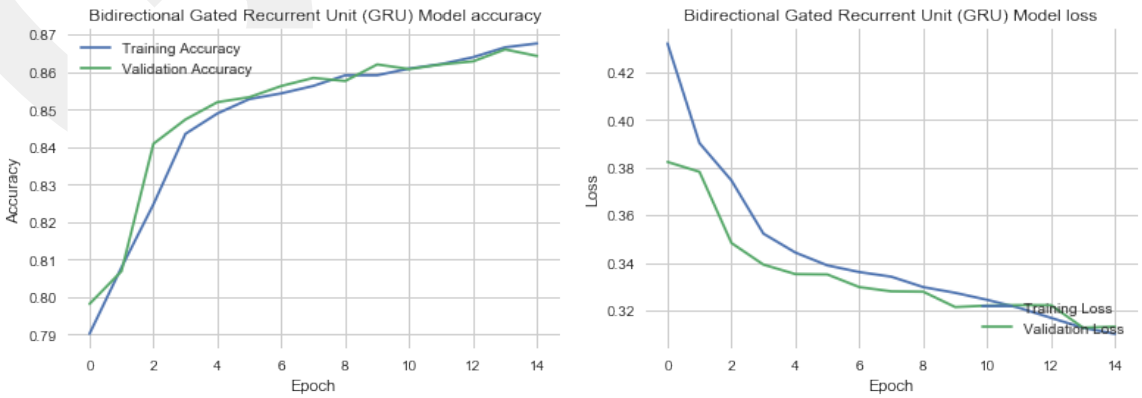

Figure 5.7: Bil-GRU Accuracy and Loss Plots

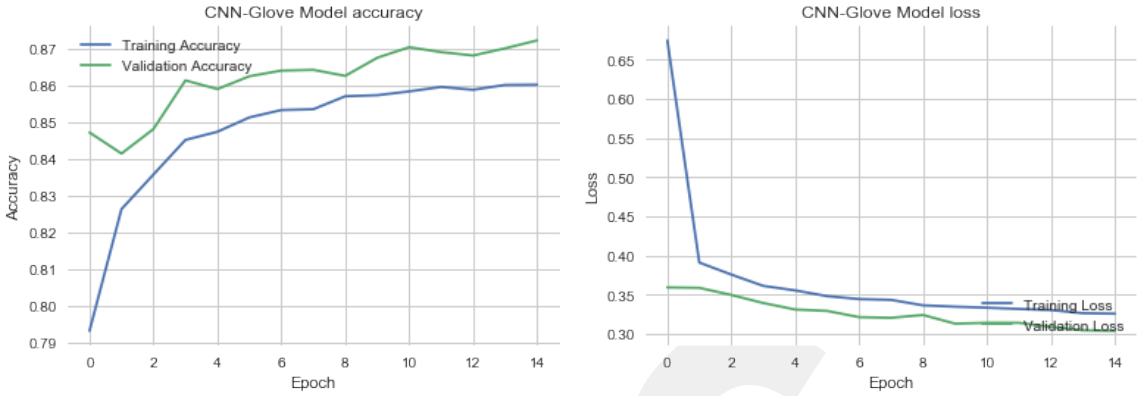

Figure 5.8: CNN with Glove Accuracy and Loss Plots

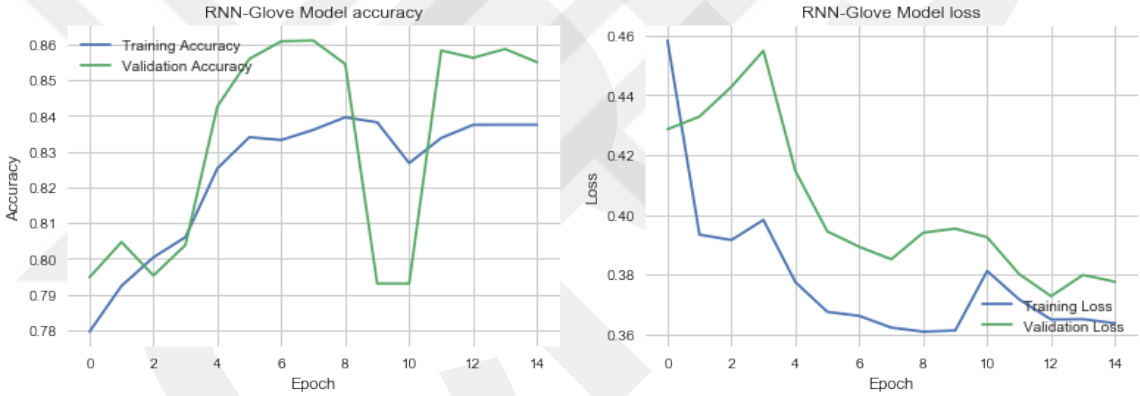

Figure 5.9: RNN with Glove Accuracy and Loss Plots

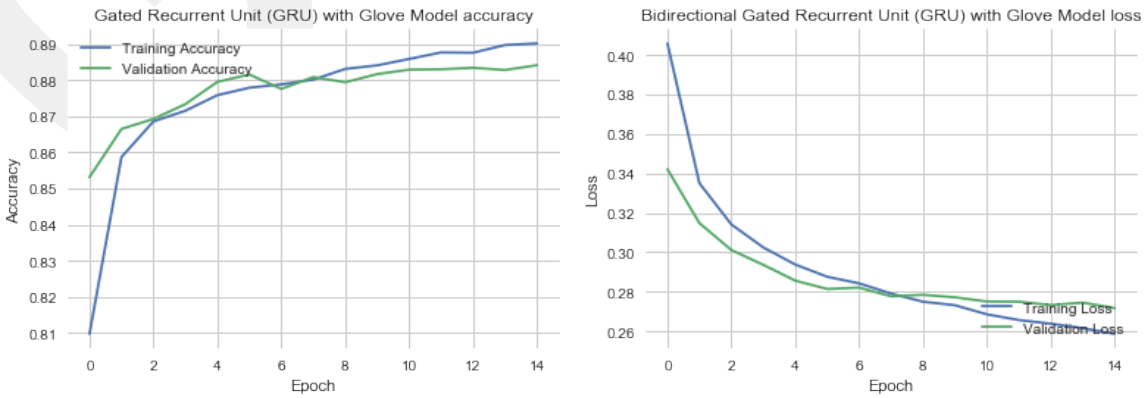

Figure 5.10: LSTM with Glove Accuracy and Loss Plots

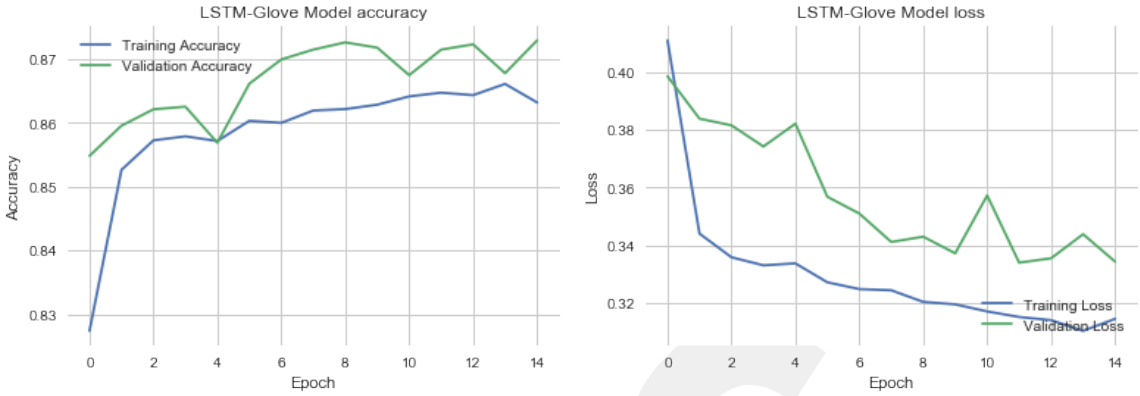

Figure 5.11: GRU with Glove Accuracy and Loss Plots

Figure 5.12: Bil-LSTM with Glove Accuracy and Loss Plots

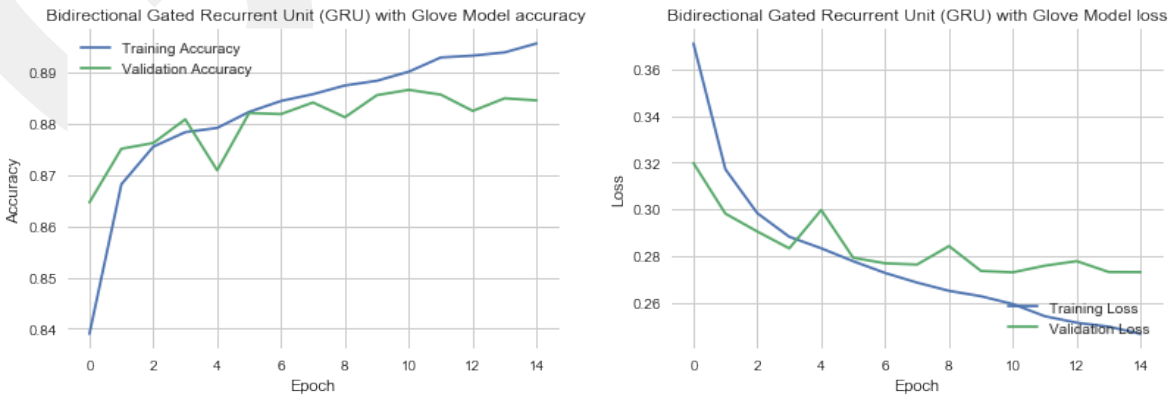

Figure 5.13: Bil-GRU with Glove Accuracy and Loss Plots

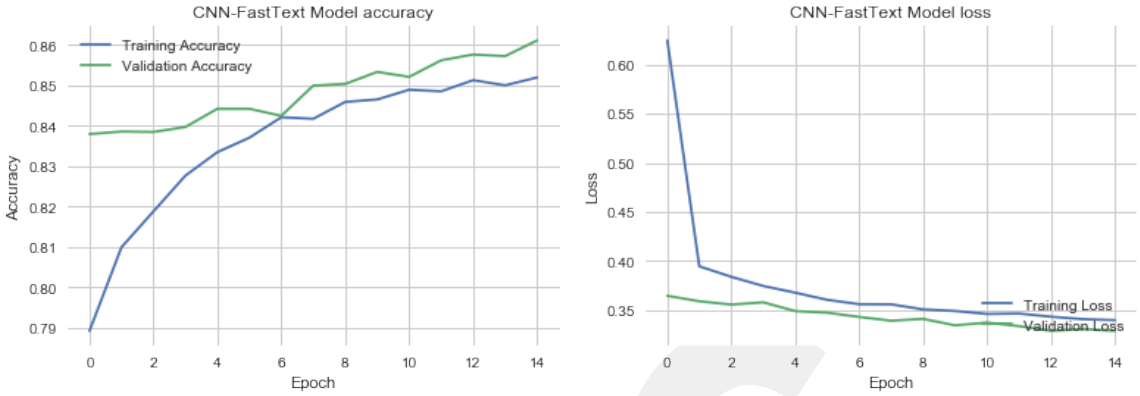

Figure 5.14: CNN with fastText Accuracy and Loss Plots

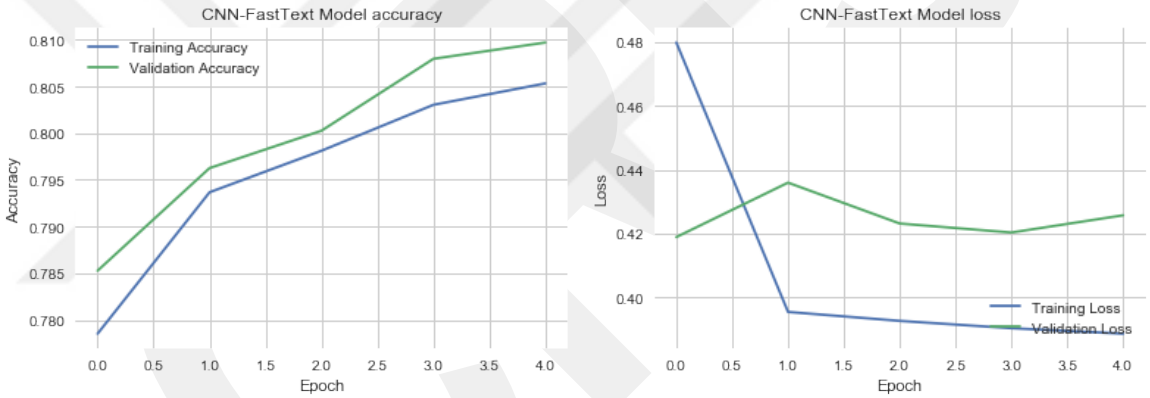

Figure 5.15: RNN with fastText Accuracy and Loss Plots

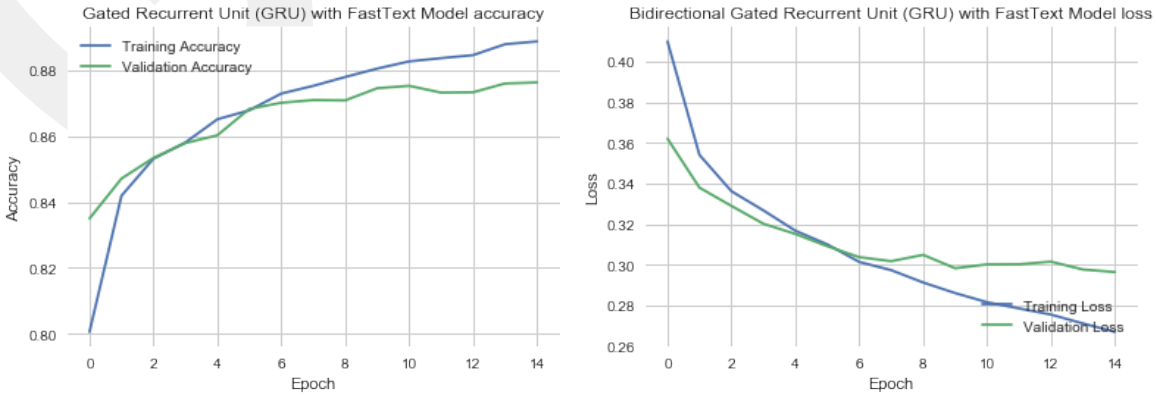

Figure 5.16: LSTM with fastText Accuracy and Loss Plots

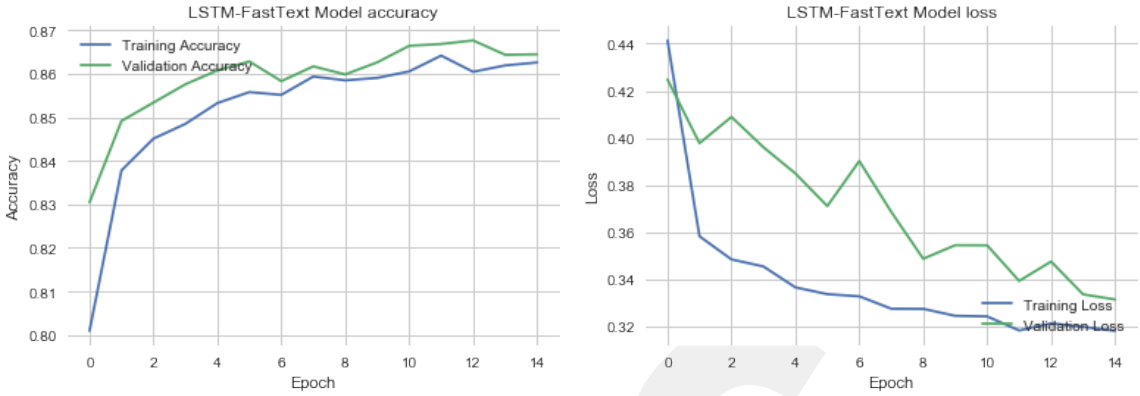

Figure 5.17: GRU with fastText Accuracy and Loss Plots

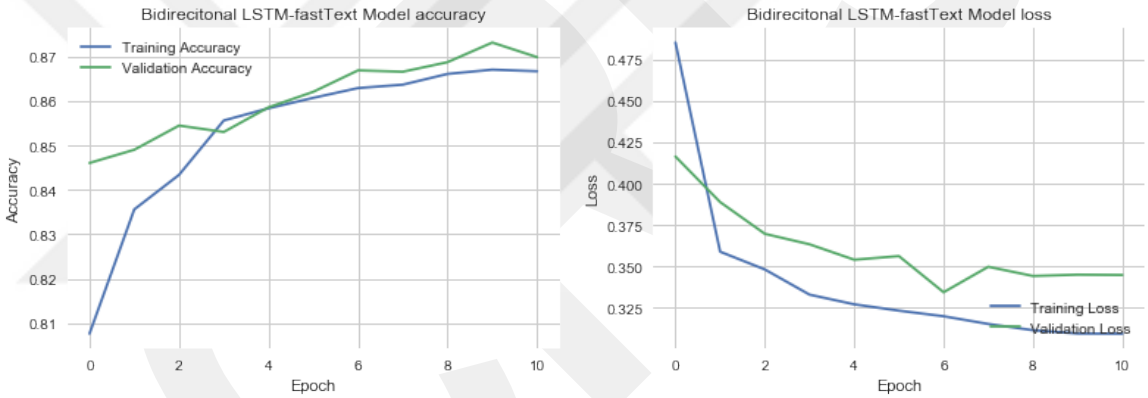

Figure 5.18: Bil-LSTM with fastText Accuracy and Loss Plots

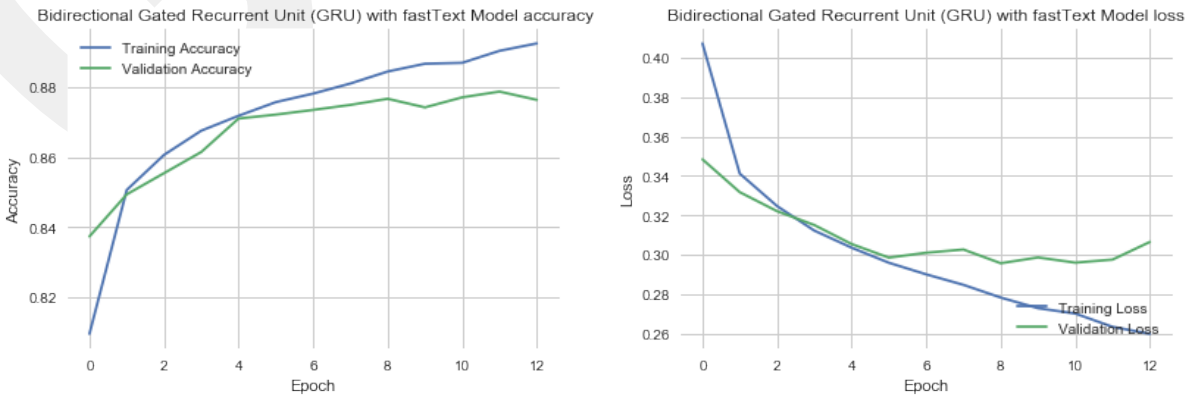

Figure 5.19: BilGRU with fastText Accuracy and Loss Plots

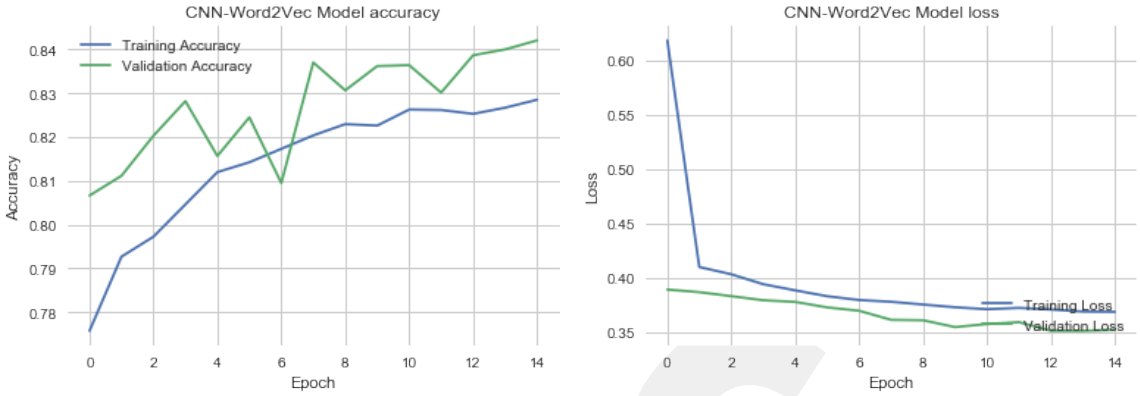

Figure 5.20: CNN with Word2Vec Accuracy and Loss Plots

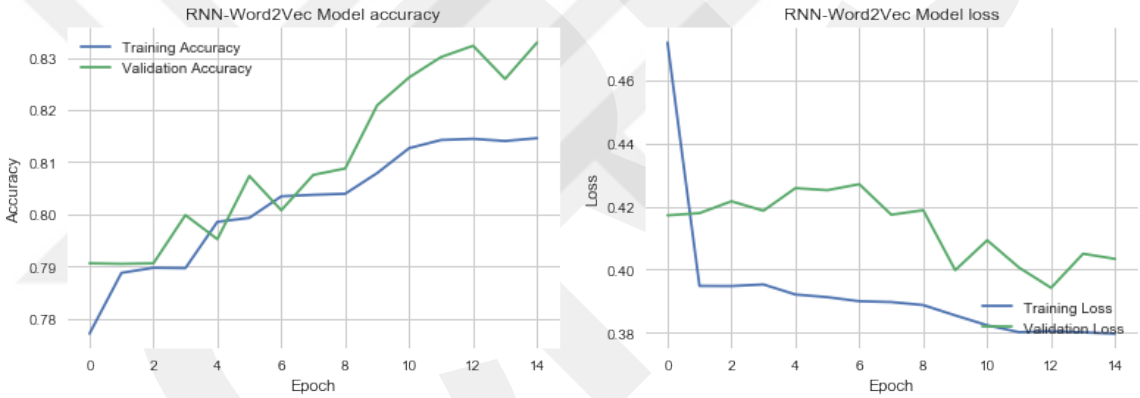

Figure 5.21: RNN with Word2Vec Accuracy and Loss Plots

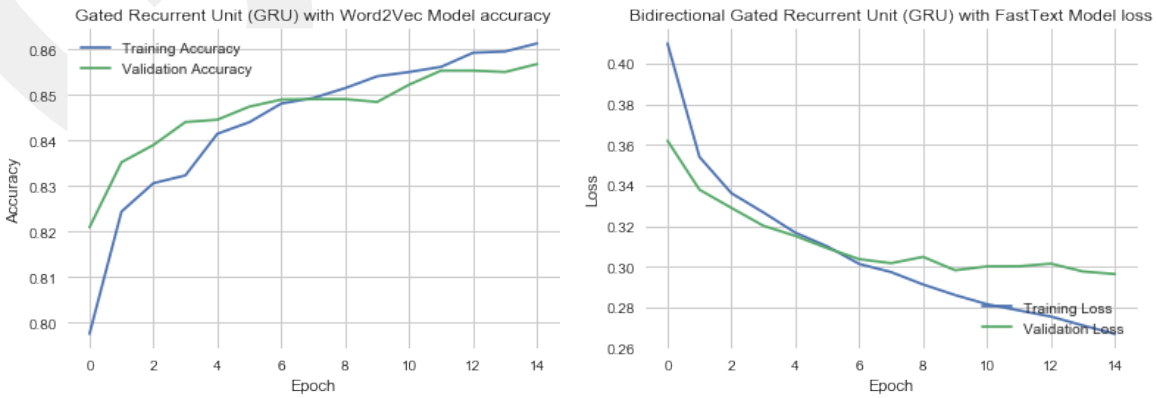

Figure 5.22: LSTM with Word2Vec Accuracy and Loss Plots

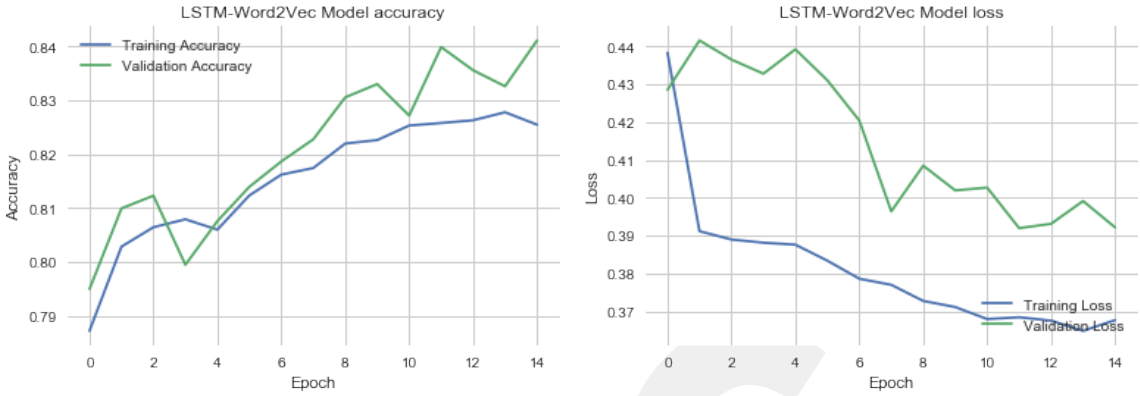

Figure 5.23: GRU with Word2Vec Accuracy and Loss Plots

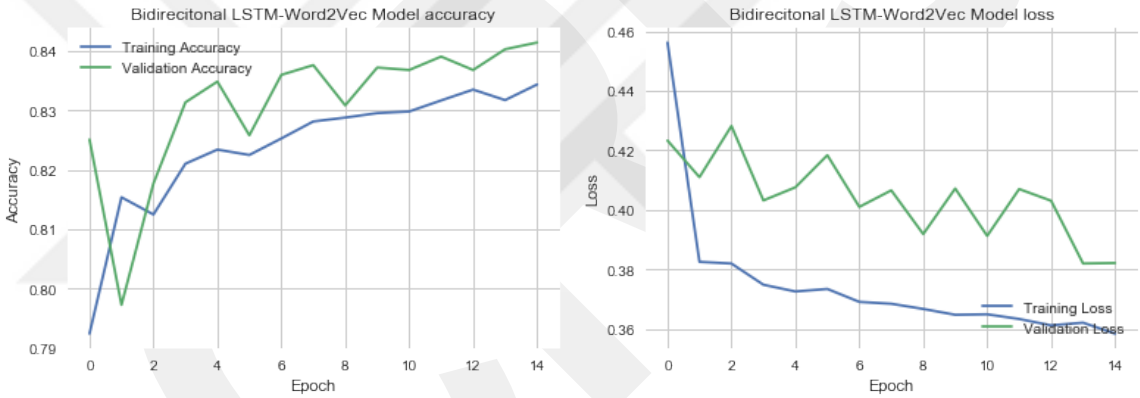

Figure 5.24: Bil-LSTM with Word2Vec Accuracy and Loss Plots

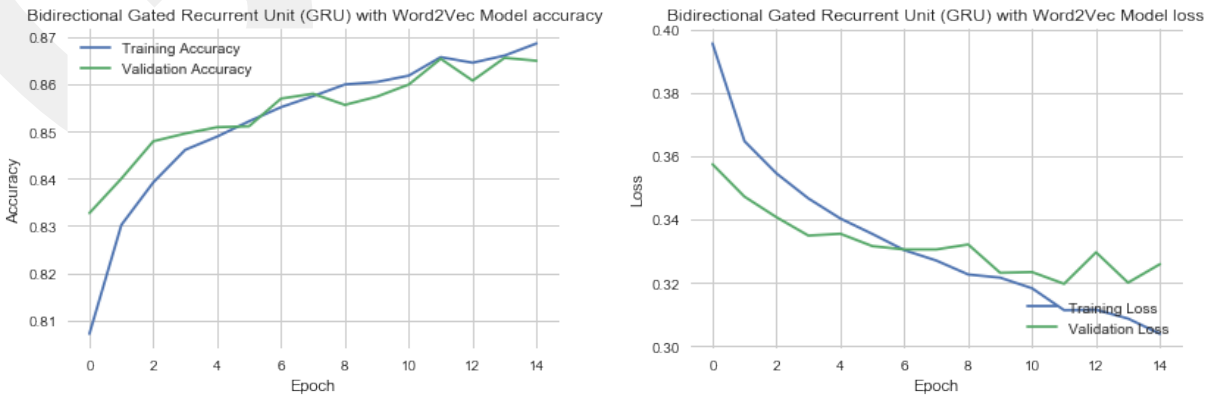

Figure 5.25: Bil-GRU with Word2Vec Accuracy and Loss Plots

CHAPTER 6

CONCLUSIONS AND FUTURE WORK

In this study, we explore the issue of multi-label classification where one instance is simultaneously classified into one or more than two classes. e explore the state-of-art trend of deep leaning methodologies, by using different set or deep neural networks. We address two approaches that vary in terms of the method of word embedding.

In the first approach, we use plain Keras library embedding layer and create a word embedding vector. We construct various architecture models such as the Neural Network Base line, the Recurrent Neural Network, the Convolutional Neural Network, the Long Short Term Memory Network, the Gated Recurrent Unit, and the Bidirectional Network (Gated Recurrent Unit, Long Short Term Memory).

In the second methodology, we utilise different word embedding in our deep learning models as a pre-trained embedded corpus. These include fastText, glove, and word2vec. We observe the behavior and conduct of these models in terms of the first and the second approaches. Finally, we compare our result with respect to previous study. We observe that even though data set is an unbalanced, the F1-measure in our studies achieve higher measures than the compared result.

The experiments demonstrate that, among the proposed methods, pre-trained word embedding significantly improves the F1-measurement relative to the other multi-label classification metrics in our models.

The study can be extended by working on developing an algorithm and technique that can handle imbalance data. Starting point can be exploring Synthetic Minority Oversampling Technique use in multi-class problem. The technique randomly picked

a point from the minority class group and calculated the k-nearest neighbors for this point. The synthetic points are added between the point selected and the neighbors. We can also apply undersampling technique to the predominance classes. The neural network model will pay more attention to the minor class samples in this method. We believe these two methods will provide improvement to the models measure that involve an unbalance data set.

REFERENCES

[1] Martın Abadi et al. “Tensorflow: A system for large-scale machine learning”. In: *12th {USENIX} Symposium on Operating Systems Design and Implementation ({OSDI} 16)*. 2016, pp. 265–283.

[2] Betty van Aken et al. “Challenges for toxic comment classification: An in-depth error analysis”. In: *arXiv preprint arXiv:1809.07572* (2018).

[3] Ben Athiwaratkun, Andrew Gordon Wilson, and Anima Anandkumar. “Probabilistic fasttext for multi-sense word embeddings”. In: *arXiv preprint arXiv: 1806.02901* (2018).

[4] Yoshua Bengio et al. “A neural probabilistic language model”. In: *Journal of machine learning research* 3.Feb (2003), pp. 1137–1155.

[5] Kush Bhatia et al. “Sparse local embeddings for extreme multi-label classification”. In: *Advances in neural information processing systems*. 2015, pp. 730–738.

[6] SN Bharath Bhushan and Ajit Danti. “Classification of text documents based on score level fusion approach”. In: *Pattern Recognition Letters* 94 (2017), pp. 118–126.

[7] Tiago Carneiro et al. “Performance Analysis of Google Colaboratory as a Tool for Accelerating Deep Learning Applications”. In: *IEEE Access* 6 (2018), pp. 61677–61685.

[8] Yun Chen et al. “Multi-label Text Classification with Deep Neural Networks”. In: *2018 International Conference on Network Infrastructure and Digital Content (IC-NIDC)*. IEEE. 2018, pp. 409–413.

[9] Everton Alvares Cherman et al. “Lazy multi-label learning algorithms based on mutuality strategies”. In: *Journal of Intelligent & Robotic Systems* 80.1 (2015), pp. 261–276.

[10] Amanda Clare and Ross D King. “Knowledge discovery in multi-label phenotype data”. In: *European Conference on Principles of Data Mining and Knowledge Discovery*. Springer. 2001, pp. 42–53.

[11] Ronan Collobert and Jason Weston. “A unified architecture for natural language processing: Deep neural networks with multitask learning”. In: *Proceedings of the 25th international conference on Machine learning*. ACM. 2008, pp. 160–167.

[12] Francesco De Comité, Rémi Gilleron, and Marc Tommasi. “Learning multi-label alternating decision trees from texts and data”. In: *International Workshop on Machine Learning and Data Mining in Pattern Recognition*. Springer. 2003, pp. 35–49.

[13] Jingcheng Du et al. “ML-Net: multi-label classification of biomedical texts with deep neural networks”. In: *arXiv preprint arXiv:1811.05475* (2018).

[14] Passent El Kafrawy, Amr Mausad, and Heba Esmail. “Experimental comparison of methods for multi-label classification in different application domains”. In: *International Journal of Computer Applications* 114.19 (2015), pp. 1–9.

[15] Yoav Goldberg and Omer Levy. “word2vec Explained: deriving Mikolov et al.’s negative-sampling word-embedding method”. In: *arXiv preprint arXiv: 1402.3722* (2014).

[16] Antonio Gulli and Sujit Pal. *Deep Learning with Keras*. Packt Publishing Ltd, 2017.

[17] Mai Ibrahim, Marwan Torki, and Nagwa El-Makky. “Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning”. In: *2018 17th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE. 2018, pp. 875–878.

[18] Rie Johnson and Tong Zhang. "Effective use of word order for text categorization with convolutional neural networks". In: *arXiv preprint arXiv:1412.1058* (2014).

[19] Armand Joulin et al. "Bag of Tricks for Efficient Text Classification". In: *arXiv preprint arXiv:1607.01759* (2016).

[20] Armand Joulin et al. "FastText.zip: Compressing text classification models". In: *arXiv preprint arXiv:1612.03651* (2016).

[21] Yoon Kim. "Convolutional neural networks for sentence classification". In: *arXiv preprint arXiv:1408.5882* (2014).

[22] Diederik P Kingma and Jimmy Ba. "Adam: A method for stochastic optimization". In: *arXiv preprint arXiv:1412.6980* (2014).

[23] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. "Imagenet classification with deep convolutional neural networks". In: *Advances in neural information processing systems*. 2012, pp. 1097–1105.

[24] Angeliki Lazaridou, Nghia The Pham, and Marco Baroni. "Combining language and vision with a multimodal skip-gram model". In: *arXiv preprint arXiv:1501.02598* (2015).

[25] Siyuan Li. "Application of recurrent neural networks in toxic comment classification". PhD thesis. UCLA, 2018.

[26] Jingzhou Liu et al. "Deep learning for extreme multi-label text classification". In: *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. ACM. 2017, pp. 115–124.

[27] Weiyang Liu et al. "Large-margin softmax loss for convolutional neural networks." In: *ICML*. Vol. 2. 3. 2016, p. 7.

[28] Helmut Lütkepohl. *Handbook of matrices*. Vol. 1. Wiley Chichester, 1996.

[29] Gjorgji Madjarov et al. "An extensive experimental comparison of methods for multi-label learning". In: *Pattern recognition* 45.9 (2012), pp. 3084–3104.

[30] Tomas Mikolov et al. *Computing numeric representations of words in a high-dimensional space*. US Patent 9,037,464. May 2015.

[31] Tomas Mikolov et al. "Distributed representations of words and phrases and their compositionality". In: *Advances in neural information processing systems*. 2013, pp. 3111–3119.

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

[33] Pushparaja Murugan. "Implementation of deep convolutional neural network in multi-class categorical image classification". In: *arXiv preprint arXiv: 1801.01397* (2018).

[34] Michael A Nielsen. *Neural networks and deep learning*. Vol. 25. Determination press San Francisco, CA, USA: 2015.

[35] Arjun Pakrashi, Derek Greene, and Brian MacNamee. "Benchmarking multi-label classification algorithms". In: *24th Irish Conference on Artificial Intelligence and Cognitive Science (AICS'16), Dublin, Ireland, 20-21 September 2016*. CEUR Workshop Proceedings. 2016.

[36] Jeffrey Pennington, Richard Socher, and Christopher Manning. "Glove: Global vectors for word representation". In: *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*. 2014, pp. 1532–1543.

[37] Yashoteja Prabhu and Manik 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*. ACM. 2014, pp. 263–272.

[38] Harry Pratt et al. "Convolutional neural networks for diabetic retinopathy". In: *Procedia Computer Science* 90 (2016), pp. 200–205.

[39] Jesse Read et al. "Classifier chains for multi-label classification". In: *Machine learning* 85.3 (2011), p. 333.

[40] Juan D Rodriguez, Aritz Perez, and Jose A Lozano. "Sensitivity analysis of k-fold cross validation in prediction error estimation". In: *IEEE transactions on pattern analysis and machine intelligence* 32.3 (2009), pp. 569–575.

[41] Sebastian Ruder. "An overview of gradient descent optimization algorithms". In: *arXiv preprint arXiv:1609.04747* (2016).

[42] Bhargav Srinivasa-Desikan. *Natural Language Processing and Computational Linguistics: A practical guide to text analysis with Python, Gensim, spaCy, and Keras*. Packt Publishing Ltd, 2018.

[43] Christian Szegedy et al. "Going deeper with convolutions". In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2015, pp. 1–9.

[44] *Toxic Comment Classification Challenge*. URL: `https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge`.

[45] Petra Vidnerova and Roman Neruda. "Evolving keras architectures for sensor data analysis". In: *2017 Federated Conference on Computer Science and Information Systems (FedCSIS)*. IEEE. 2017, pp. 109–112.

[46] Yao-Yuan Yang et al. "Deep learning with a rethinking structure for multi-label classification". In: *arXiv preprint arXiv:1802.01697* (2018).

[47] Zichao Yang et al. "Hierarchical attention networks for document classification". In: *Proceedings of the 2016 conference of the North American chapter of the association for computational linguistics: human language technologies*. 2016, pp. 1480–1489.

[48] Chih-Kuan Yeh et al. "Learning deep latent space for multi-label classification". In: *Thirty-First AAAI Conference on Artificial Intelligence*. 2017.

[49] Ian En-Hsu Yen et al. "Pd-sparse: A primal and dual sparse approach to extreme multiclass and multilabel classification". In: *International Conference on Machine Learning*. 2016, pp. 3069–3077.

[50] *Your Home for Data Science*. URL: `https://www.kaggle.com/`.

[51] Liang-Chih Yu et al. “Refining word embeddings for sentiment analysis”. In: *Proceedings of the 2017 conference on empirical methods in natural language processing*. 2017, pp. 534–539.

[52] Ana Zelaia et al. “A multiclass/multilabel document categorization system: Combining multiple classifiers in a reduced dimension”. In: *Applied Soft Computing* 11.8 (2011), pp. 4981–4990.

[53] Min-Ling Zhang and Zhi-Hua Zhou. “ML-KNN: A lazy learning approach to multi-label learning”. In: *Pattern recognition* 40.7 (2007), pp. 2038–2048.

[54] Min-Ling Zhang and Zhi-Hua Zhou. “Multilabel neural networks with applications to functional genomics and text categorization”. In: *IEEE transactions on Knowledge and Data Engineering* 18.10 (2006), pp. 1338–1351.

[55] Wenjie Zhang et al. “Deep extreme multi-label learning”. In: *Proceedings of the 2018 ACM on International Conference on Multimedia Retrieval*. ACM. 2018, pp. 100–107.

[56] Xiang Zhang, Junbo Zhao, and Yann LeCun. “Character-level convolutional networks for text classification”. In: *Advances in neural information processing systems*. 2015, pp. 649–657.

APPENDIX A

NOMENCLATURE

L	Set of labels
d	instance toxic text comment
l	label
σ	sigma
z_t	update gate
z_t	reset gate
x_t	input vector
i_t	input gate
f_t	forget gate
z_t	reset gate
σ_t	output gate activation vector.
C_t	cell state
h_t	output vector of LSTM cell unit.
V	matrices
W	matrices

Table A.1: Nomenclature Table

APPENDIX B

DATA SET DESCRIPTION

id	*comment_text*	*toxic*	*severe_toxic*	*obscene*	*threat*	*insult*	*identity_hate*
0036e50f 42d0b679	Oh, it's me vandalising?xD See here. Greetings,	0	0	0	0	0	0
0036621 e4c7e10b57	Would you both shut up, you don't run wikipedia, especially a stupid kid.	1	1	1	0	1	0
003217c3 eb469ba9	Hi! I am back again! Last warning! Stop undoing my edits or die!	1	0	0	0	1	0
0007e25b 2121310b	Bye! Don't look, come or think of comming back! Tosser.	1	0	0	0	0	0
0020e711 9b96eeeb	Stupid peace of shit stop deleting my stuff asshole go die and fall in a hole go to hell!	1	1	1	0	1	0

Table B.1: Train Data Description

id	***comment_text***
0000247867823ef7	== From RfC == The title is fine as it is, IMO.
00017695ad8997eb	I don't anonymously edit articles at all.
0001ea8717f6de06	Thank you for understanding. I think very highly of you and would not revert without discussion
0002f87b16116a7f	":::: Somebody will invariably try to add Religion? Really?? You mean, the way people have invariably kept adding ""Religion"" to the Samuel Beckett infobox? And why do you bother bringing up the lo...
0016b94c8b20ffa6	I WILL BURN YOU TO HELL IF YOU REVOKE MY TALK PAGE ACCESS!!!!!!!!!!!!
001d739c97bc2ae4	How dare you vandalize that page about the HMS Beagle! Don't vandalize again, demon!

Table B.2: Test Data Description

id	*toxic*	*severe_toxic*	*obscene*	*threat*	*insult*	*identity_hate*
0000247867823ef7	-1	-1	-1	-1	-1	-1
00017695ad8997eb	-1	-1	-1	-1	-1	-1
0001ea8717f6de06	-1	-1	-1	-1	-1	-1
0002f87b16116a7f	-1	-1	-1	-1	-1	-1
0016b94c8b20ffa6	-1	-1	-1	-1	-1	-1
001d739c97bc2ae4	-1	-1	-1	-1	-1	-1

Table B.3: Test Labels Data Description