KWAME NKRUMAH UNIVERSITY OF SCIENCE AND TECHNOLOGY

COLLEGE OF SCIENCE

MINI PROJECT

DOCUMENTATION

Supervisor: Dr. Kwabena Owusu-Agyemang

Name: Fiasorgbor George Etornam Kobla
Index No: 9404419

Email Spam Detection using BERT

Contents

LIST OF FIGURES

LIST OF TABLES

CHAPTER ONE

INTORDUCTION

1.1 GENERAL OVERVIEW

Online communication has largely replaced face-to-face interaction in our daily lives as a result of the increased use of social networks and the Internet. Email is one of the most popular forms of formal and corporate communication due of its open access, speed, and dependability. Spam emails, which are one or more unsolicited messages that take the shape of advertising or promotional materials like debt relief plans, get rich fast schemes, online dating, and health-related products, etc., were rapidly expanding as email usage increased. Automatic spam detection is not a novel concept; businesses and organizations are constantly seeking for methods to enhance user experience. To safeguard their computers against damage in the event that spam emails contained viruses. Additionally, it prevents the waste of time and network resources (bandwidth). In an effort to overcome this issue, the NLP and ML communities have contributed a number of SPAM detection data sets and trained models to categorize entire documents. Depending on the type of data and task they are given, several categorization algorithms may perform worse or better. This has been accomplished by utilizing several NLP techniques, beginning with deep learning's contribution to natural language understanding. Convolutional neural network (CNN) for categorizing sentences. For sequence tagging, use BiLSTMattention-based bidirectional LSTM networks for topic-based sentiment analysis and relational categorization. resulting in the most recent transfer learning models, where research was concentrated on utilizing trained models for various NLP tasks. In comparison to all other works, the BERT language model has the highest outcomes. When doing word embedding, BERT considers the extensive left and right semantic contexts of words and can produce various semantic representations for the same word depending on context. In this research, BERT based Model is investigated for the task of automatic spam detection. The system is evaluated against different novel approaches on a publicly available corpus of spam and ham emails.

1.1.1 BACKGROUND

The most widely used communication methods today are emails and text messages, and as the number of people using these tools rises, so does the prevalence of spam. Spam is any form of bulk, unsolicited, unwanted digital communication. Spam emails and SMSs waste a lot of resources by unnecessarily flooding network lines. Although most spam emails come from marketers wanting to promote their goods, some are much more malevolent in nature, such as phishing emails that try to deceive recipients into disclosing personal information like website login credentials or credit card numbers. Phishing is a type of cybercrime. In order to combat spam, numerous studies and efforts are made to create spam detectors that can identify messages and emails as spam or ham. In this research, we developed a spam detector utilizing the BERT pre-trained model, which categorizes emails and messages based on their context. We also trained our spam detector model to recognize spam and ham messages.

1.2 PROBLEM STATEMENT

Every day, there is a fierce battle between spammers and filtering techniques since spammers have started employing cunning strategies to get past the spam filters. The development of machine learning models that can forecast the behavior of these spammers is not given enough attention. Since the user must go through the undesirable junk mail and it uses up storage space and communication bandwidth, spam is a time waster for the user. To make it harder for spammers to trick spam filters, rules in other existing filters must be continuously updated and maintained. Substituting vocables with synonyms or similar words makes the message unrecognizable to the algorithms used to detect spam as one tactic used to fool spam filters. Businesses and people have lost a lot of money and suffered significant suffering as a result of spam. There have been introduced and developed a number of models and ways to automatically detect spam emails, but none of them demonstrated 100% predictive accuracy. Both machine learning and deep learning algorithms outperformed all other proposed models. The models' accuracy was improved via natural language processing (NLP). This study introduces the usefulness of word embedding for categorizing spam emails. The work of separating spam emails from legitimate emails is carried out by a pre-trained transformer model called BERT (Bidirectional Encoder Representations from Transformers).

1.3 AIMS AND OBJECTIVES

AIMS:

To develop a BERT (Bidirectional Encoder Representations from Transformers) model-based spam detection program to be able to detect spam messages no matter how hard they are designed to beat spam filters.

OBJECTIVES:

i. To study how to use machine learning techniques for spam detection.

ii. To study how to incorporate and train models to perform tasks.

iii. To leverage modified machine learning algorithm in knowledge analysis software.

iv. To study different ways to develop more effective spam detection softwares.

1.4 JUSTIFICATION

Spam filtering implementation is crucial for any firm. Spam filtering not only helps keep junk out of email inboxes, but it also improves the functionality and usefulness of business communications by ensuring that they are only used for what they were intended for. Since many email-based attacks attempt to deceive users into clicking on a malicious file, asking them for their credentials, and other information, spam filtering is essentially an anti-malware tool.

Radicati Research Group Inc. estimates that email spam costs organizations up to $20.5 billion annually, and that amount will only go up. Spam filtering stops these spam messages from ever arriving in an inbox, preventing businesses from contributing to the rising number of lost sales. Without spam filters, an organization's email system wouldn't work correctly, and the chance of a cyber-attack on internal data would be increased.

1.5 RESEARCH QUESTIONS

1. How effective is a spam filter.
2. What issue can be found with spam filters.
3. Why is spam filtering important.
4. How do spam filters determine which emails are likely to be spam?

1.6 LIMITATIONS OF MY STUDY

The largest drawback of utilizing an email filter is the possibility that messages could be mistakenly classified as spam due to an error in the algorithm. Even with the strongest spam filters available, mails might still end up with incorrect labels, claims Steven Scott, a Bayesian specialist.

While it's annoying to miss out on crucial emails, we also need to consider the possibility that you might miss the same emails if you receive a lot of spam. Given that hundreds of emails are sent out each day, how are you supposed to see that message from the boss? Even if you pay close attention, you could miss certain emails.

1.7 SCOPE OF THE PROJECT

This project needs a coordinated scope of work. These scopes will help to focus on this project. The scopes are:

i. Modified existing machine learning algorithm.

ii. Make use and classify of a data set including data preparation, classification and visualization.

iii. Score of data to determine the accuracy of spam detection

1.8 PROJECT LIMITATION

i. This project can only detect and calculate the accuracy of spam messages only.

ii. It focuses on filtering, analyzing and classifying the messages.

iii. Does not block the messages, it only determines if the message is spam or ham.

iv. May block harmless messages.

v. Needs constant updating and maintance.

1.9 SUMMARY

The project's background, problem statement, objectives, scope, and work limitations are all covered in Chapter 1 of this handbook. I can conclude from all of these that this project uses the BERT model to identify spam emails.

CHAPTER TWO

LITERATURE REVIEW

2.1 INTRODUCTION

In this chapter, the literature review for the machine learning classifiers utilized in earlier studies and projects is covered. It does not involve information collecting but rather a summary of earlier studies pertinent to this topic. Searching, reading, analyzing, summarizing, and assessing the reading materials in accordance with the project are all part of the process.

Literature reviews on the subject of machine learning have revealed that the majority of spam filtering and detection systems require periodic training and updating. To begin functioning, spam filters also need rules to be established. As a result, the user will soon find it burdensome.

2.2 MACHINE LEARNING

Existing machine learning algorithms are used in this project and adjusted to meet its requirements. The algorithms used in machine learning are effective in reviewing enormous amounts of data. Because of the growing amount of processed data, it usually becomes better with time. It provides the system additional practice, which can be utilized to produce more accurate forecasts.

Without the need for human involvement, machine learning enables immediate adaptation. It recognizes emerging threats and trends and puts the necessary safeguards in place. Due to its automated nature, it also saves time.

2.2.1 SPAM DETECTION

The overall approach and tools used to execute the spam email detection task are described in detail. Generally, any NLP task consists of five main phases: data collection, data pre-processing, feature extraction, model training, and model evaluation. Fig. 1 shows the flow for those phases. Hence, in this work, feature extraction will be done automatically as part of the deep learning model training,

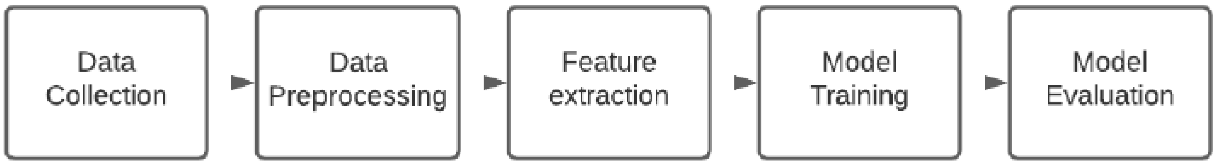


Fig. 1. Genearl Flow of Main Phases for NLP Task

Data Collection Tow open-source data sets were used in this work, both have two columns, the body text of the email and the class label spam or ham. The first data set is the open source Spambase data set from the UCI machine learning repository, the data set contains 5569 emails, of which 745 are spam. The second data set is the opensource Spam filter data set from Kaggle which contains 5728 emails of which 1368 are spam. Exploring the distribution for SPAM and HAM classes, both data sets are imbalanced; where the SPAM class is the rare class. In order to avoid biasing for the major class which is the HAM class, a new balanced training data set is created through merging spam samples that are randomly selected from the two data sets Spambase and Spam filter with ensuring of no duplicate records. The new data set contains 2000 samples of SPAM and 3000 samples of HAM. In addition, hold-out portion from Spambase data set was reserved for testing the persistence of the model against unseen samples. The task is a binary-classification problem to detect whether the text is classified as SPAM 1 or HAM 0. Table 2 shows the distribution for both classes HAM and SPAM in both the training and testing data sets.

Data Cleaning and Pre-Processing After going through the class distribution for each target, the distribution of the word number in each record is explored. SPAM texts are longer than HAM texts in general, the input sequence for our model in this case study is defined to be 300 tokens or words. The next step was to clean the input data by extracting and removing the stop words using the Sklearn library, since stop words provide little to no unique information that can be used for classification, then punctuation marks were extracted and removed as they affect the text encoding part, especially when it is attached with a word, for e.g: [users, user’s] have different encoding. Text kept in cased shape as it might be an indicator for SPAM, especially promotion scams or other types of Spams. Keras tokenization tool used to split the words as tokens based on the space. For example, [“We went to Amman.”] it will be tokenized as the following: [‘We’, ‘went’, ’to’, ’Amman.’]. Each token got encoded to vector for classical classifiers using TfidfVectorizer from Keras. The label target variable is encoded to a binary format SPAM 1 and HAM 0.

Base Line Model The baseline model is A state-of-the-art BiLSTM model. It takes the input from an embedding layer and its output vectors are fed to a Dense layer. The Dense layer uses Relu activation function to allows converging quickly . The next layer is a Dropout layer of 0.1 to avoid overfitting. Last layer is a Dense layer that uses Sigmoid activation function to normalize the output. Fig. 2 illustrate the baseline model structure and its layers. In addition to BiLSTM model, a set of classic classifiers KNN with n-neighbors=3 and MultinomialNB are trained to compare results with.

Transformer Model BERT transformer is a pre-trained model published by Google AI language. It uses attention models to learn the contextual relation between the words in a sentence. It mainly consists of two parts: an encoder that encodes the input text and a decoder for the output result based on the task. The bert-base-cased model implemented using Simple Transformers library which is an API built above Hugging face library. Bert cased model has been trained on English Wikipedia with 2500 million words and BookCorpus with 800 million words.The model was trained using hyper-parameters: 300 sequence length, three epochs, 32 batch size, 4e-5 learning rate, and AdamW as an optimizer.

2.3 RELATED WORK

Spam e-mails detection problem has already drawn researchers' attention. Several significant works to detect spam e-mails have been proposed. In this section; prior related works that focus on the spam classification using ML and deep learning techniques are discussed.

Thiago S. Guzella et. Al (2009) conducted "A Review of Machine Learning Approaches to Spam Filtering". In their paper, they found that Bayesian Filters that are used to filter spam required a long training period for it to learn before it can completely well function.

S. Ananthi (2009), conducted research on "Spam Filtering using K-NN". In this paper, she used KNN as it is one of the simplest algorithms. An object is classified by a majority vote of its neighbours where the class is typically small.

Anjali Sharma et. Al (2014) has conducted "A Survey on Spam Detection Techniques". In this paper, they found that Artificial Neural Network (ANN) must be trained first to categorize emails into spam or non-spam starting from the particular data sets.

Simon Tong and Daphne Koller (2001), has conducted research on "Support Vector Machine Active Learning with Applications to Text Classification". In this paper, they presented new algorithm for active learning with SVM induction and transduction. It is used to reduce version space as much as it can at every query. They found out that the existing dataset only differ by one instance from the original labelled data set.

Minoru Sasaki and Hiroyuki Shinnou (2005) has conducted research on "Spam Detection Using Text Clustering". They used text clustering based on vector space model to construct a new spam detection technique. This new spam detection model can find spam more efficiently even with various kinds of mail.

Aigars Mahinovs and Ashutosh Tiwari (2007) has conducted research on “Text Classification Method Review”. They test the process of text classification using different classifier which is natural language processing, statistical classification, functional classification and neural classification. They found that all the classifier works well but need more improvement especially to the feature preparation and classification engine itself in order to optimize the classification performance.

2.4 SUMMARY

This chapter discussed the technique and model used in the proposed project. The method and model are chosen based on the previous research articles and journals. Some advantages and disadvantages of each model are also discussed.

CHAPTER THREE

METHODOLOGY

3.1 INTRODUCTION

This chapter will explain the specific details on the methodology being used in order to develop this project. Methodology is an important role as a guide for this project to make sure it is in the right path and working as well as plan. There is different type of methodology used in order to do spam detection and filtering. So, it is important to choose the right and suitable methodology thus it is necessary to understand the application functionality itself.

3.2 IMPLEMENTATION AND CODING

This project is developed by using Python Language and combining with the implementation of the BERT model. Google Collaboratory is the platform used to develop the project. It contains important functions for preprocessing the dataset. Then, the dataset is going to be used to train and test the machine learning model and achieve the objectives of the project.

3.3 PROJECT REQUIREMENT AND SPECIFICATION

System requirements are needed in order to accomplish the project goals and objectives and to assist in development of the project. That involves the usage of hardware and software. Each of these requirements is related to each other to make sure that system can be developed smoothly.

3.3.1 HARDWARE

Table 1 The hardware used in completing this project are as follows;

No.	Hardware	Type	Description
1.	Laptop	HP Pavilion	Processor: Intel(R) Core(TM) i5-6200U CPU @ 2.30GHz 2.40 GHz OS version: Windows 64 bit RAM: 8 GB

2.	Printer	HP Deskjet	Printing document
3.	Printed paperwork	A4 paper	Used to study on how to implement this project from past paperwork

3.3.2 SOFTWARE

The software used in completing this project are as follows;

No.	Software	Description
1.	Google Collaboratory	Machine learning platform Deploy models Run models in cloud
2.	Tensorflow_hub	Place where all TensorFlow pre-trained models are stored.
3.	Tensorflow	For model creation
4.	Pandas	For data loading, manipulation and wrangling.
5.	Tensorflow_text:	Allows additional NLP text processing capabilities outside the scope of tensorflow.
6.	Skelarn	For doing data splitting
7.	Matplotlib	For visualization support
8.	Microsoft Word 2016	Creating and editing report
9.	Microsoft PowerPoint 2016	For presenting finding and result of the project
10.	Github	Get dataset
11.	Snipping Tool	Captures and screenshot images

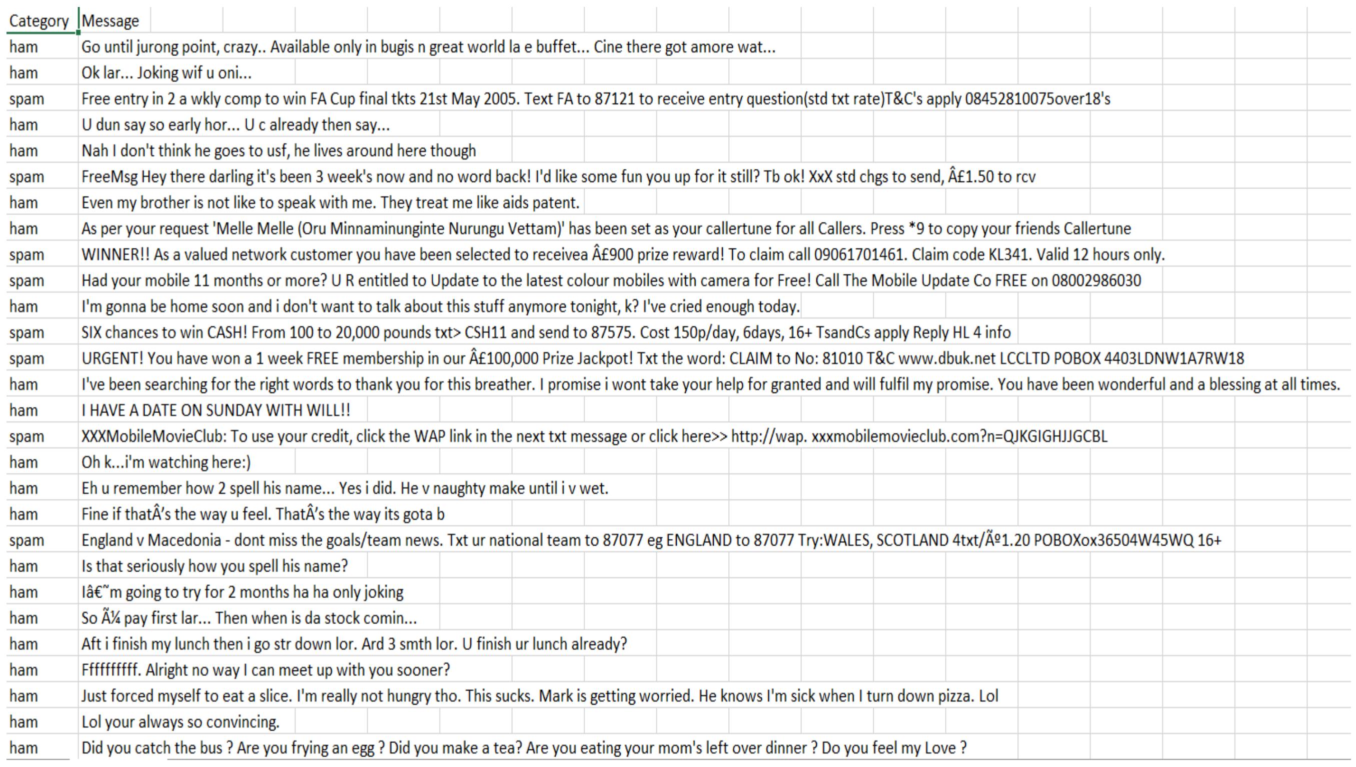
3.4 FRAMEWORK

3.4.1 DATA SOURCE

Due to several restrictions, such as the volume of data and the throughput needed for accurate and timely intake, gathering data is incredibly challenging. The dataset I utilized for this project is genuine data that can be downloaded from a site that houses machine learning data.

3.4.2 DATA SETS

To start we will first download a readily available dataset. The data first need to be reformatted into CSV to make it easier to use for further process. Once you downloaded it will look something like this:



Category	Message
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, Â£1.50 to rcv
ham	Even my brother is not like to speak with me. They treat me like aids patent.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
spam	WINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
spam	URGENT! You have won a 1 week FREE membership in our Â£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.
ham	I HAVE A DATE ON SUNDAY WITH WILL!!
spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
ham	Oh k...i'm watching here:)
ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
ham	Fine if thatÂ's the way u feel. ThatÂ's the way its gota b
spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/Ãº1.20 POBOXox36504W45WQ 16+
ham	Is that seriously how you spell his name?
ham	Iâ€˜m going to try for 2 months ha ha only joking
ham	So Ã¼ pay first lar... Then when is da stock comin...
ham	Aft i finish my lunch then i go str down lor. Ard 3 smth lor. U finish ur lunch already?
ham	Fffffffff. Alright no way I can meet up with you sooner?
ham	Just forced myself to eat a slice. I'm really not hungry tho. This sucks. Mark is getting worried. He knows I'm sick when I turn down pizza. Lol
ham	Lol your always so convincing.
ham	Did you catch the bus ? Are you frying an egg ? Did you make a tea? Are you eating your mom's left over dinner ? Do you feel my Love ?

Figure 2 Categorized Dataset

3.5 The Pipeline Overview for Spam Detection Using BERT

1. Load Data – I will be loading the data which is simple [2 categories (ham and spam) along with corresponding emails] CSV file.

2. EDA – Perform some EDA to get a feel of what data looks like

3. Pre-Processing – Based on the results of EDA we will be going to apply some preprocessing to the data to make it model friendly

4 – Model Creation – We will use **Functional API*** to build our Deep Learning Model which will consist of

- **INPUT –** Will create an input layer that will take in all the pre-processing data and pass it to bert_processor.
- **BERT PREPROCESSOR –** Process our input text to be in the format which encoder accepts. (Adds MASK AND ENCODINGS).
- **BERT ENCODER –** Feed our processed text to the bert model which will eventually generate a contextualized word embedding for our training data.
- **DROPOUT –** Add a dropout layer that will randomly drop out a few neurons from the network to take care overfitting problem.
- **SOFTMAX –** At last we will add a softmax layer to predict data in 2 categories.
- + 2 more generic steps.

5- Model Evaluation – Further we will check how the model performs on the test data and plot some relevant charts and metrics based on the observations.

6- Predict Data – Finally we will predict our own emails using the model.

Here is a general visualization of the flow of logic:

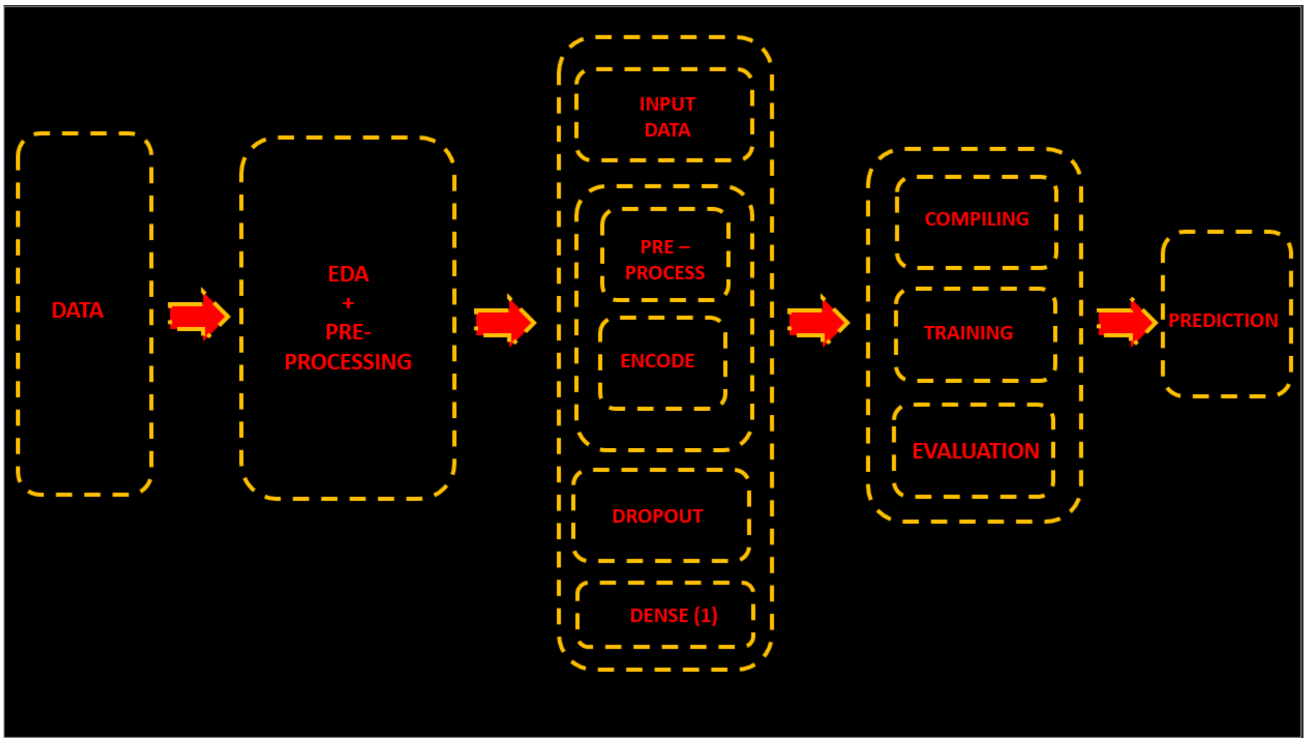


Figure 3 Outline

3.6 SUMMARY

One of the most crucial aspects of system and application development is methodology. Any type of application can be created using one of the many different software development methodologies that are currently accessible. The project can be completed inside the allotted time frame with the aid of the proper methods. The activities in each phase of the technique are described in detail for simple comprehension.

CHAPTER FOUR

IMPLEMENTATION AND RESULT

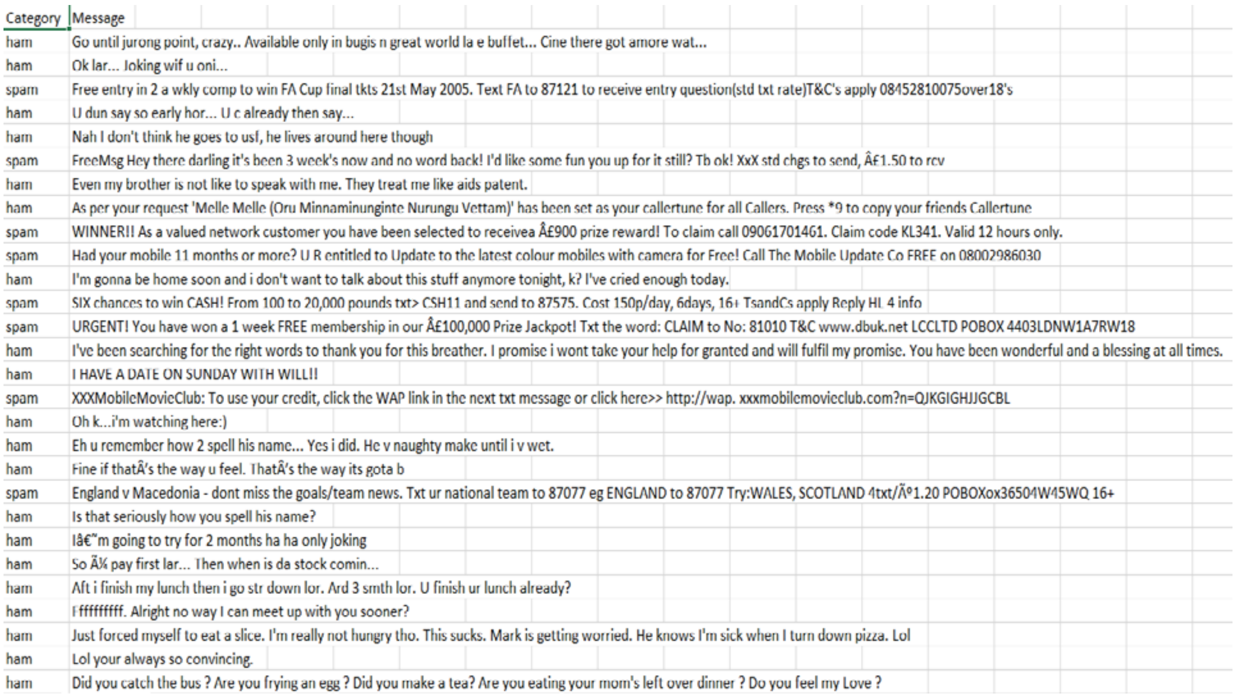
4.1 INTRODUCTION

In this chapter, I will discuss the project implementation and testing. The project's final development stage includes both implementation and testing results. The trained model's project development must be implemented in order to confirm that it complies with the requirement. The process of displaying the results of the testing that was done to make sure it was functional is known as testing result. The machine learning model will demonstrate good functionality at this stage, and any shortcomings will be identified for future improvement. Thus, after the development of the full project, this chapter will normally describe the deployment, testing, and implementation of the project.

4.2 IMPLIMENTATION

Implementing Spam Detection Using BERT

To start we will first download a readily available dataset. Once you downloaded it will look something like this:



Category	Message
ham	Go until jurong point, crazy.. Available only in bugis n great world la e buffet... Cine there got amore wat...
ham	Ok lar... Joking wif u oni...
spam	Free entry in 2 a wkly comp to win FA Cup final tkts 21st May 2005. Text FA to 87121 to receive entry question(std txt rate)T&C's apply 08452810075over18's
ham	U dun say so early hor... U c already then say...
ham	Nah I don't think he goes to usf, he lives around here though
spam	FreeMsg Hey there darling it's been 3 week's now and no word back! I'd like some fun you up for it still? Tb ok! XxX std chgs to send, Â£1.50 to rcv
ham	Even my brother is not like to speak with me. They treat me like aids patent.
ham	As per your request 'Melle Melle (Oru Minnaminunginte Nurungu Vettam)' has been set as your callertune for all Callers. Press *9 to copy your friends Callertune
spam	WINNER!! As a valued network customer you have been selected to receivea Â£900 prize reward! To claim call 09061701461. Claim code KL341. Valid 12 hours only.
spam	Had your mobile 11 months or more? U R entitled to Update to the latest colour mobiles with camera for Free! Call The Mobile Update Co FREE on 08002986030
ham	I'm gonna be home soon and i don't want to talk about this stuff anymore tonight, k? I've cried enough today.
spam	SIX chances to win CASH! From 100 to 20,000 pounds txt> CSH11 and send to 87575. Cost 150p/day, 6days, 16+ TsandCs apply Reply HL 4 info
spam	URGENT! You have won a 1 week FREE membership in our Â£100,000 Prize Jackpot! Txt the word: CLAIM to No: 81010 T&C www.dbuk.net LCCLTD POBOX 4403LDNW1A7RW18
ham	I've been searching for the right words to thank you for this breather. I promise i wont take your help for granted and will fulfil my promise. You have been wonderful and a blessing at all times.
ham	I HAVE A DATE ON SUNDAY WITH WILL!!
spam	XXXMobileMovieClub: To use your credit, click the WAP link in the next txt message or click here>> http://wap. xxxmobilemovieclub.com?n=QJKGIGHJJGCBL
ham	Oh k...i'm watching here:)
ham	Eh u remember how 2 spell his name... Yes i did. He v naughty make until i v wet.
ham	Fine if thatÂ's the way u feel. ThatÂ's the way its gota b
spam	England v Macedonia - dont miss the goals/team news. Txt ur national team to 87077 eg ENGLAND to 87077 Try:WALES, SCOTLAND 4txt/Ãº1.20 POBOXox36504W45WQ 16+
ham	Is that seriously how you spell his name?
ham	Iâ€˜m going to try for 2 months ha ha only joking
ham	So Ã¼ pay first lar... Then when is da stock comin...
ham	Aft i finish my lunch then i go str down lor. Ard 3 smth lor. U finish ur lunch already?
ham	Ffffffffff. Alright no way I can meet up with you sooner?
ham	Just forced myself to eat a slice. I'm really not hungry tho. This sucks. Mark is getting worried. He knows I'm sick when I turn down pizza. Lol
ham	Lol your always so convincing.
ham	Did you catch the bus ? Are you frying an egg ? Did you make a tea? Are you eating your mom's left over dinner ? Do you feel my Love ?

Figure 4 Dataset

Here ham – Good Mails and Spam – Spam mails. Our job will be to build a model which can train on the data and return prediction when given new inputs.

Loading Dependencies

So, first of all, we are going to import few libraries namely:

- **Tensorflow_hub**: Place where all TensorFlow pre-trained models are stored.
- **Tensorflow:** For model creation
- **Pandas:** For data loading, manipulation and wrangling.
- **Tensorflow_text**: Allows additional NLP text processing capabilities outside the scope of tensorflow.
- **Skelarn:** For doing data splitting
- **Matplotlib**: For visualization support

```
import tensorflow_hub as hub

import pandas as pd

import tensorflow_text as text

import matplotlib.pyplot as plt

from sklearn.model_selection import train_test_split

import tensorflow as tf

import numpy as np
```

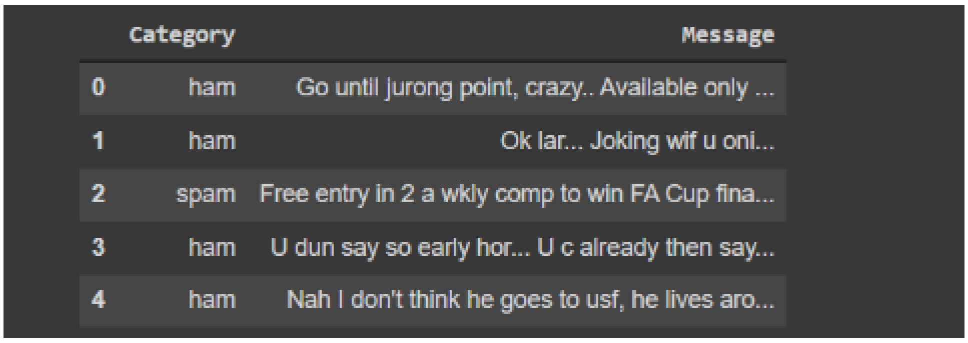
Loading Data

Now we will just load our data into a pandas dataframe ‘**df**‘ using its **read_csv()** method/fn and view the
first 5 samples using the **head()** method:

```
# load data
df = pd.read_csv('/content/spam_data.csv')
```

```
df.head()
```

Executing the above code returns a data frame with the first 5 samples:



	Category	Message
0	ham	Go until jurong point, crazy.. Available only ...
1	ham	Ok lar... Joking wif u oni...
2	spam	Free entry in 2 a wkly comp to win FA Cup fina...
3	ham	U dun say so early hor... U c already then say...
4	ham	Nah I don't think he goes to usf, he lives aro...

Spam Detection Exploratory Data Analysis

We will now do some of the Exploratory – Data Analysis to check how data is distributed along two categories. This will give us a feel if we need to do some type of preprocessing over data or is it on the same scale.

To perform this operation we will just be **grouping the data based on category** and call **value_counts()**
method on it like:

# check count and unique and top values and their frequency
df['Category'].value_counts()
This returns **no of samples** of each class as:

```
ham     4825
spam     747
Name: Category, dtype: int64
```

Clearly, the data is imbalanced and there are more good emails(ham) than spam emails. This may lead to a problem as a model may learn all the features of the ham emails over spam emails and thus always predict all emails as ham (OVERFITTIN!). So before proceeding, we need to take care of that.

Downsampling Data

Downsampling is a technique where the majority class is downsampled to match the minority class. Since our data has only one column(feature) it ok to use it.

We perform downsampling by just picking any random 747 samples from the ham class. Here is the code
to do that:

1) We first calculated the percentage of data that needs to be balanced by **dividing minority (spam) by**
majority(ham):

```
# check percentage of data - states how much data needs to be balanced
print(str(round(747/4825,2))+'%')
```

```
>> 0.15%
```

It came out to be 15% of data that needs to be balanced.

2) We then create 2 new datasets namely ham and spam and filtered the data having categories as spam

and ham and append it to the respective dataset and finally printed their shape to confirm the filtering and
creation:)

```
# creating 2 new dataframe as df_ham , df_spam df_spam = df[df['Category']=='spam'] df_ham =
df[df['Category']=='ham'] print("Ham Dataset Shape:", df_ham.shape) print("Spam Dataset Shape:",
```

df_spam.shape)

```
>> Spam Dataset Shape: (747, 2)
   Ham Dataset Shape: (4825, 2)
```

Seems like filtering and separation were successful.

3) Now we will sample the ham dataset using the sample () method with the shape of our spam dataset and
to be more specific load it into a new dataframe **df_ham_downsampled and** print its shape to cross verify.

```
# downsampling ham dataset - take only random 747 example # will use df_spam.shape[0] - 747
df_ham_downsampled = df_ham.sample(df_spam.shape[0]) df_ham_downsampled.shape
>> (747, 2)
```

4) Finally, we will **concatenate** our **df_ham_downsampled** and **df_spam** to create a final dataframe called **df_balalnced.**

```
# concating both dataset - df_spam and df_ham_balanced to create df_balanced dataset df_balanced =pd.concat([df_spam , df_ham_downsampled])
```

Checking the value counts again projects downsampling we have done-
df_balanced['Category'].value_counts()

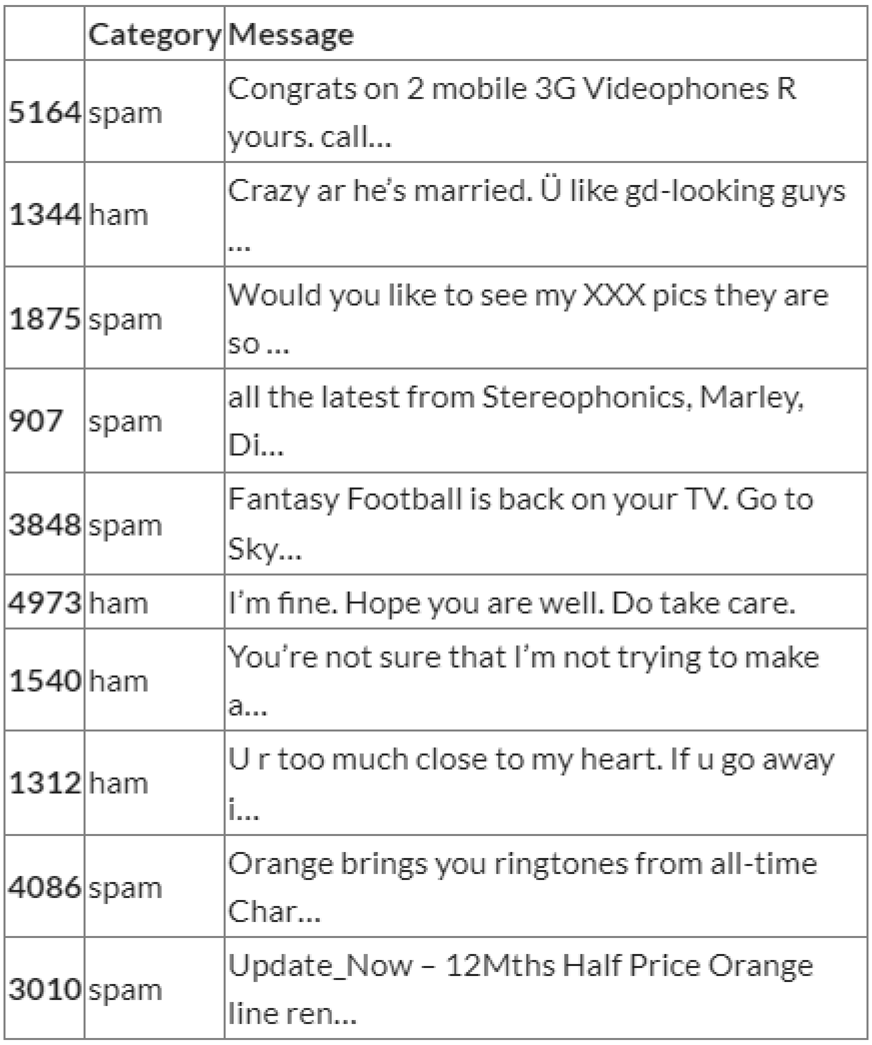
Returns:

```
>> spam 747
ham 747
Name: Category, dtype: int64
```

Printing few samples wouldn’t hurt either – 10 samples

Df balanced.sample(10)

Table 2 Returns – Evenly distributed data of spam and ham (no in cade of head)



	Category	Message
5164	spam	Congrats on 2 mobile 3G Videophones R yours. call…
1344	ham	Crazy ar he's married. Ü like gd-looking guys …
1875	spam	Would you like to see my XXX pics they are so …
907	spam	all the latest from Stereophonics, Marley, Di…
3848	spam	Fantasy Football is back on your TV. Go to Sky…
4973	ham	I'm fine. Hope you are well. Do take care.
1540	ham	You're not sure that I'm not trying to make a…
1312	ham	U r too much close to my heart. If u go away i…
4086	spam	Orange brings you ringtones from all-time Char…
3010	spam	Update_Now – 12Mths Half Price Orange line ren…

Preprocessing of Spam Detection Data

One Hot Encoding Categories

As can be seen, we have only text as categorical data, and the model doesn't understand them. So instead of text, we can just assign integer labels to our class **ham and spam as 0 and 1 respectively,** and store it in new column **spam. This is called- Hot-Encoding**

To achieve this we will just be filtering the column category and perform operations:

- 1 – If a category is a ham/ not spam
- 0 – if the category is spam

The best part is that we can use the **one-liner lambda function** to achieve the following result and **apply** it to the dataframe for all values.

Lambda Fn Syntax = [lambda x : value expression else value]

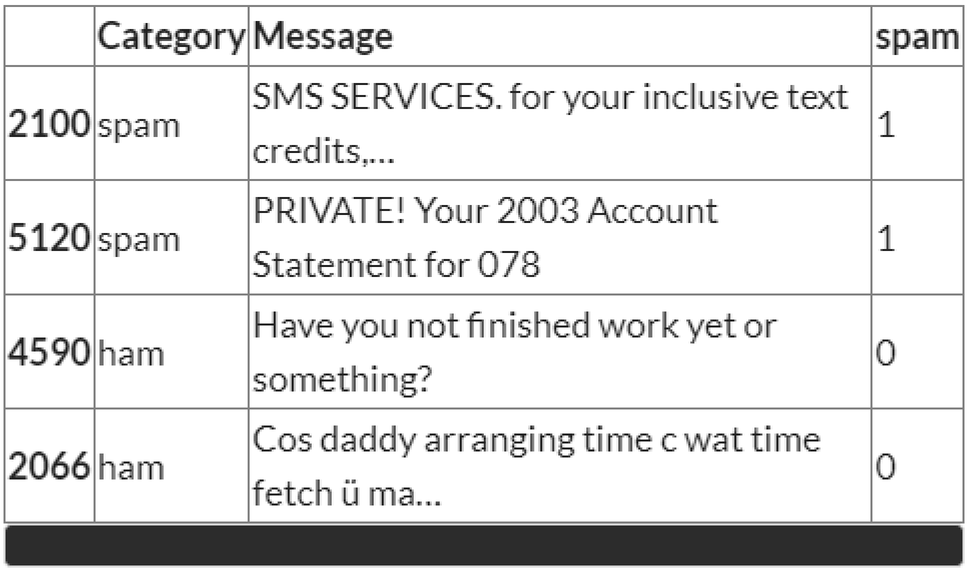
So we can use the following code to achieve the desired result:

# creating numerical repersentation of category - one hot encoding df_balanced['spam'] = df_balanced['Category'].apply(lambda x:1 if x=='spam' else 0)

For checking the result we can now just print few samples as :

# displaying data - spam -1 , ham-0 df_balanced.sample(4)

Returns :



	Category	Message	spam
2100	spam	SMS SERVICES. for your inclusive text credits,...	1
5120	spam	PRIVATE! Your 2003 Account Statement for 078	1
4590	ham	Have you not finished work yet or something?	0
2066	ham	Cos daddy arranging time c wat time fetch ü ma...	0

Modification to the data frame was successful.

Performing Train Test Split

Now as our data is processed, we can feed it to the model, but if we do so it may be that model will learn

the patterns of the data, and when we evaluate it will always predict the right results, which leads to biasing of the model. So we will follow the train test strategy.

Train Test Split Strategy:

It states out of all data (population) a large amount of data goes in for training(about80%) which has inputs(Messages) and Labels(1/,0) and the remaining 20% will not be fed to the model. When we further do evaluation we can just predict on that 20% and see how it

performs as the model will not be biased now.

* Note :- it is generally considered to split data as 80:20 as good ratio but can be experimented:)

The above can be achieved using the **train_test_split** fn of **sklearn.**

# loading train test split from sklearn.model_selection import train_test_split X_train, X_test , y_train, y_test = train_test_split(df_balanced['Message'], df_balanced['spam'], stratify = df_balanced['spam'])

Here we first loaded the fn from sklearn.model_selection module and performed the split and set **stratify**
to select almost equal no samples from the dataset.
Here:

- X_train, y_train – training inputs and labels – Training Set
- X_test, y_test – testing inputs and labels – Testing Set

This marks the end of the pre-processing part and now our model is ready for training. But before that, we need to generate word embedding and that’s what we are going to see in the next section.

4.3 MODEL CREATION

To create our model we will first **download the bert preprocessor and encoder**(for more info refer to the previous article) as it allows us to use them as **function pointers** where one can feed our inputs and get the **processed output and embedding**. Also, this helps in better **readability** of the code.

```
# Downloading preprocessing files and model

bert_preprocessor=hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_preprocess/3')

bert_encoder=hub.KerasLayer('https://tfhub.dev/tensorflow/bert_en_uncased_L-12_H-768_A-12/4')
```

Here:

- **bert_preprocessor –** preprocessor
- **bert_encoder –** main model (layers – 12, Hidden Layers – 768 and Attention – 12)

Creating Model

Having downloaded the bert model, we can now use Keras Functional API to build our model.

```
text_input = tf.keras.layers.Input(shape = (), dtype = tf.string, name = 'Inputs')

preprocessed_text = bert_preprocessor(text_input)

embeed = bert_encoder(preprocessed_text)

dropout = tf.keras.layers.Dropout(0.1, name = 'Dropout')(embeed['pooled_output'])

outputs = tf.keras.layers.Dense(1, activation = 'sigmoid', name = 'Dense')(dropout)

# creating final model
```

```
model = tf.keras.Model(inputs = [text_input], outputs = [outputs])
```

A couple of points to notice here: –

- **text_inpu**t: As our model data shape can be anything so we will pass shape parameters as **shape()**, its data type as **tf. string**, and name as **Inputs.**

- **dropout:-** For the dropout layer, we have set **dropout rate 0.1 –** 10% of neurons will randomly shut off and passed **embedding dictionary pooled_output** as an input to this layer i.e simply passing **entire training data embeddings** to the dropout layer.

- **outputs**: **Sigmod** as activation as the problem is a binary in nature problem, however, relu can also be used

- **model**: Our model inputs will be an **array,** so used **[]** in inputs and outputs.

Printing the model summary returns the model architecture and no of the trainable and non-trainable parameters (weights):

```
# check the summary of the model

model.summary()
```

As can be seen, there are over 109,482,241 parameters that are from the BERT model itself and are non-trainable. This gives the feel of what this LRM is.

<u>Compiling And Training Model</u>

As a generic step in the model building, we will now *compile our model* using a**dam** as our optimizer and **binary_crossentropy** as our loss function. /For metrics, we will use **accuracy, precession, recall**, and loss.

```
Metrics = [tf.keras.metrics.BinaryAccuracy(name = 'accuracy'),

          tf.keras.metrics.Precision(name = 'precision'),

          tf.keras.metrics.Recall(name = 'recall')

          ]

# Compiling our model

model.compile(optimizer ='adam',

              loss = 'binary_crossentropy',

              metrics = Metrics)
```

And now comes the best part, training our model. For training, we will just **fit the model** with the **training set** and let it run for **10 epochs** (one can experiment with it), and store the result in the history variable.

```
history = model.fit(X_train, y_train, epochs = 10)
```

Now let’s see the result which was returned:

It looks like we have a 91% accurate model and have a good precession and recalls so we can now evaluate our model.

Precision is defined as the fraction of relevant instances among all retrieved instances. **Recall** sometimes referred to as ‘sensitivity, is the fraction of retrieved instances among all relevant instances.

A perfect classifier has **precision** and **recall** both equal to **1**

Model Visualization (Neural Network Model Graph)

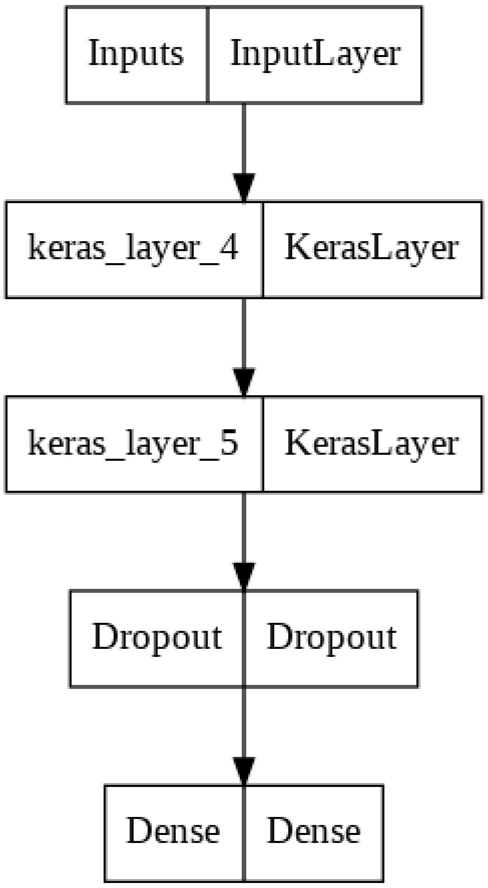


Figure 5 Plot of Neural network model graph

4.4 MODEL EVALUATION

To evaluate our model we will simply use the **model's evaluate** method feeding it **testing data and labels** to get a rough estimate of how the model is performing.

```
# Evaluating performance

model.evaluate(X_test,y_test)
```

Output

It is similar to training results which may lead to the wrong interpretation of the model. So we need a better way to understand how our model is performing and usually, classification reports and confusion matrices are the way to go.

Plotting Confusion Matrix and Classification Reports

We can use the sklearn confusion matrix and classification report which takes in actual labels(y_test) and predicted labels(y_pred) and returns an array of numbers which we will be plotting using seaborn's heatmap and matplotlib.

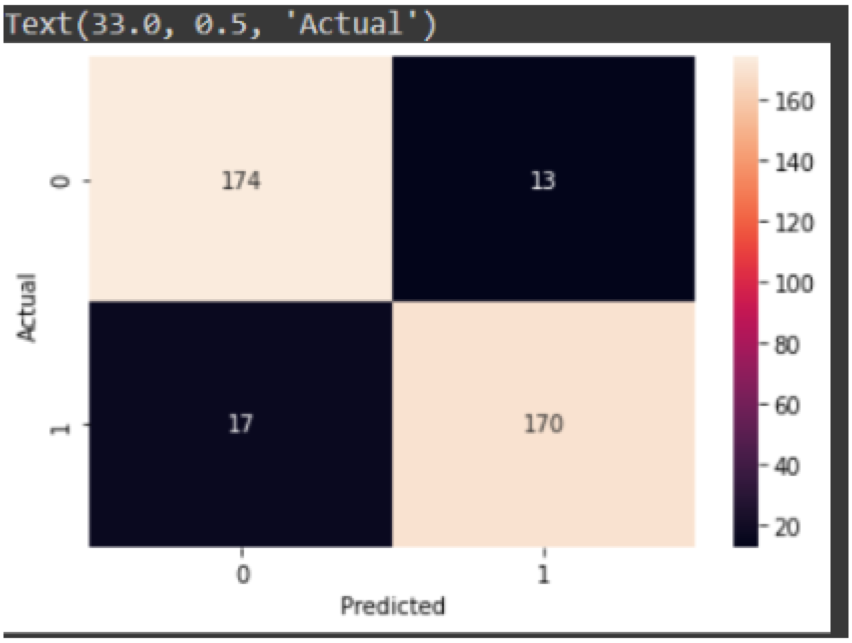
```
# getting y_pred by predicting over X_text and flattening it

y_pred = model.predict(X_test)

y_pred = y_pred.flatten() # require to be in one-dimensional array , for easy manipulation

# importing confusion maxtrix


from sklearn.metrics import confusion_matrix , classification_report



# creating confusion matrix
```

```
cm = confusion_matrix(y_test,y_pred)


cm
>> array([[174,  13],
          [ 17, 170]])
```

Here we are just predicting X_test to get **y_pred(prediction)** and **flattening** it to reshape it to a one-dimensional vector and feeding it to the **confusion _matrix function** and storing the result in **cm.**

Plotting results returns a nice looking graphs as :

```
# plotting as a graph - importing seaborn
import seaborn as sns
# creating a graph out of confusion matrix
sns.heatmap(cm, annot = True, fmt = 'd')
plt.xlabel('Predicted')
plt.ylabel('Actual')
```


```
Text(33.0, 0.5, 'Actual')
```

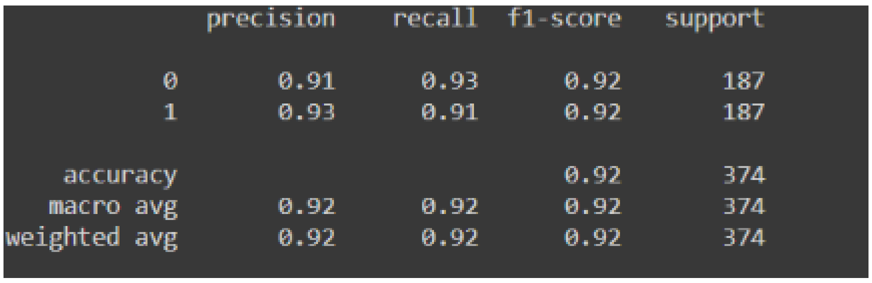
As can be seen for X-axis – We have Predicted Values, For Y-axis – We have Actual Values. Also at diagonal, we have model correct predictions.

Through the graph we can see that out of a total, 174 times the mail was ham (0) and the model predicted it right and for 170 times it was spam and model predicted spam(1), so overall we have created a good model, however one can experiment with the parameters, layers and network architecture to increase it

Classification report is also plotted similarly:

```
# printing classification report

print(classification_report(y_test , y_pred))
```

Returns:



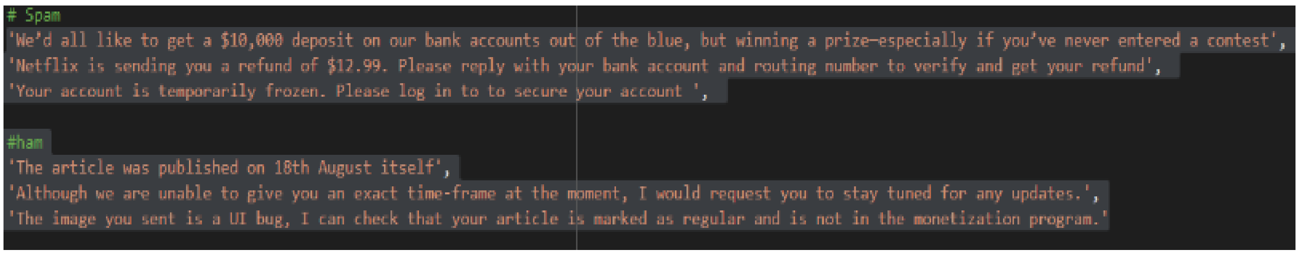
```
              precision    recall  f1-score   support

           0       0.91      0.93      0.92       187
           1       0.93      0.91      0.92       187

    accuracy                           0.92       374
   macro avg       0.92      0.92      0.92       374
weighted avg       0.92      0.92      0.92       374
```

Here also it is evident that the model is a good one as recall and accuracy us good:)

4.5 MODEL PREDICTION

Now let’s check how the model performs on real-world data.

Here I have a collection of few **spam** and **ham** emails:



```
# Spam
'We’d all like to get a $10,000 deposit on our bank accounts out of the blue, but winning a prize—especially if you’ve never entered a contest',
'Netflix is sending you a refund of $12.99. Please reply with your bank account and routing number to verify and get your refund',
'Your account is temporarily frozen. Please log in to to secure your account ',

#ham
'The article was published on 18th August itself',
'Although we are unable to give you an exact time-frame at the moment, I would request you to stay tuned for any updates.',
'The image you sent is a UI bug, I can check that your article is marked as regular and is not in the monetization program.'
```

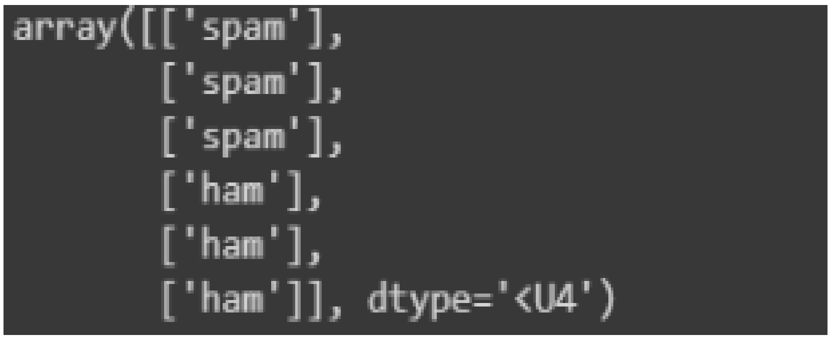
Let’s see how it performs on it:

```
predict_text = [
        # Spam
        'We’d all like to get a $10,000 deposit on our bank accounts out of the blue, but winning a prize—especially if you’ve never entered a contest',
        'Netflix is sending you a refund of $12.99. Please reply with your bank account and routing number to verify and get your refund',
        'Your account is temporarily frozen. Please log in to to secure your account ',
        #ham
        'The article was published on 18th August itself',
                'Although we are unable to give you an exact time-frame at the moment, I would request you to stay tuned for any updates.',

                'The image you sent is a UI bug, I can check that your article is marked as regular and is not in the monetization program.'

]

test_results = model.predict(predict_text)

output = np.where(test_results>0.5,'spam', 'ham')
```

Here all we have done is:

- Created a **list** that contains all the sentences.
- **Predicted** the sentence category using our **model** and stored the result in variable '**test_result**'.
- Created a **lambda function** for *filtering values greater than 'as spam' else, not 'spam'* using **NumPy. where** which ideally searches for the values and stores them in the **'outputs**'.

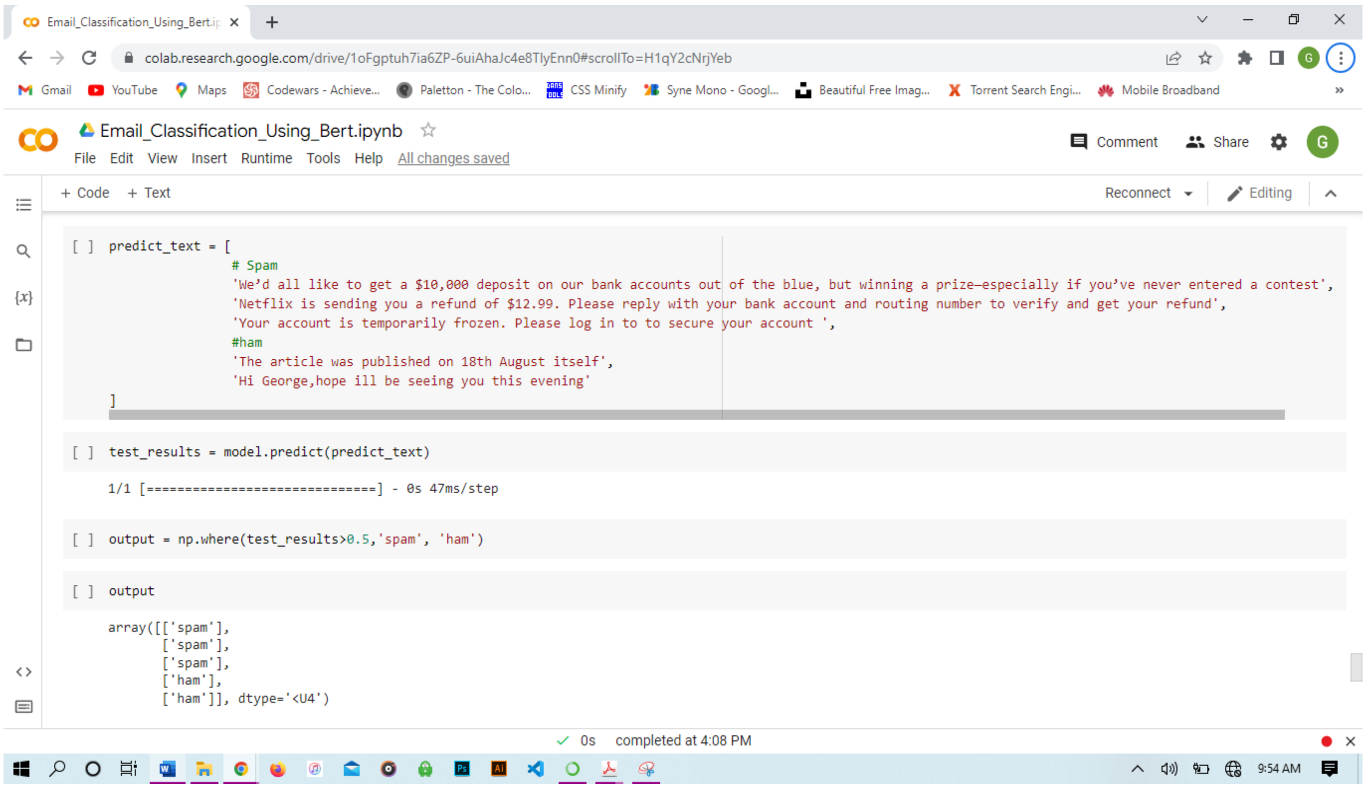
Now let's check the result:



```
array([['spam'],
       ['spam'],
       ['spam'],
       ['ham'],
       ['ham'],
       ['ham']], dtype='<U4')
```

As can be seen, it classified all our data correctly.

4.6 INTERFACE

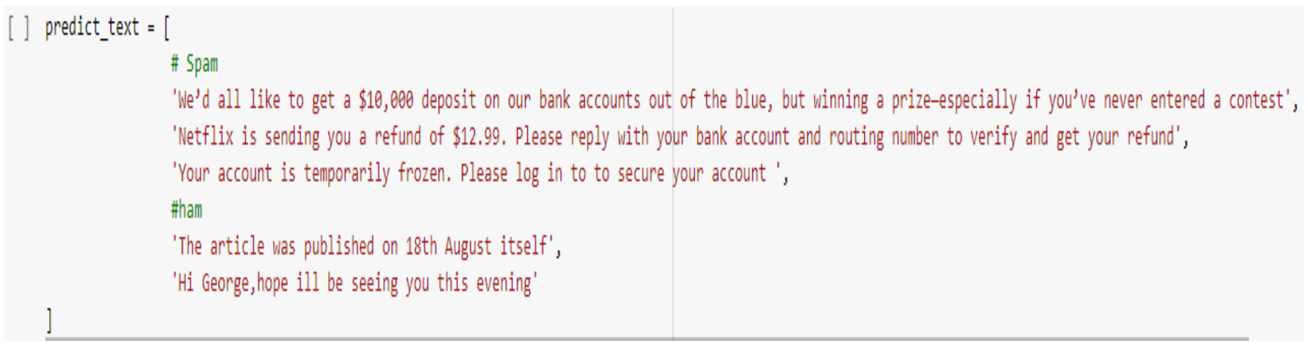

Email_Classification_Using_Bert.ipynb
colab.research.google.com/drive/1oFgptuh7ia6ZP-6uiAhaJc4e8TlyEnn0#scrollTo=H1qY2cNrjYeb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Reconnect
Editing

```
[ ] predict_text = [
                # Spam
                'We'd all like to get a $10,000 deposit on our bank accounts out of the blue, but winning a prize—especially if you've never entered a contest',
                'Netflix is sending you a refund of $12.99. Please reply with your bank account and routing number to verify and get your refund',
                'Your account is temporarily frozen. Please log in to to secure your account ',
                #ham
                'The article was published on 18th August itself',
                'Hi George,hope ill be seeing you this evening'
    ]

[ ] test_results = model.predict(predict_text)

    1/1 [==============================] - 0s 47ms/step

[ ] output = np.where(test_results>0.5,'spam', 'ham')

[ ] output

    array([['spam'],
           ['spam'],
           ['spam'],
           ['ham'],
           ['ham']], dtype='<U4')
```

Figure 6 Interface

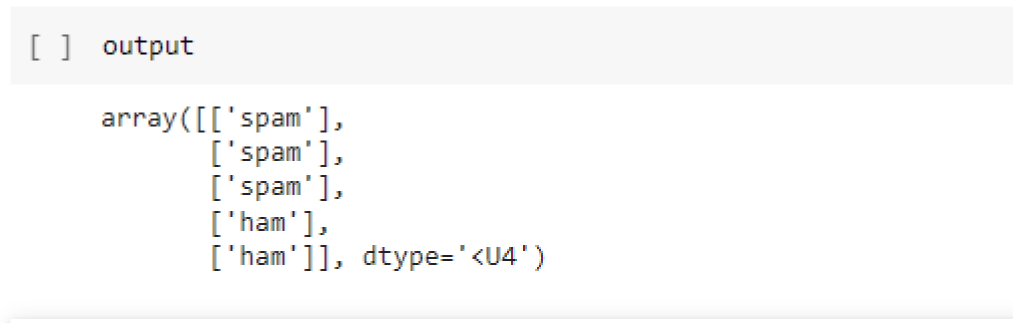
This is the program interface on google colab.

The email messages to be evaluated are;



```
[ ] predict_text = [
                # Spam
                'We'd all like to get a $10,000 deposit on our bank accounts out of the blue, but winning a prize—especially if you've never entered a contest',
                'Netflix is sending you a refund of $12.99. Please reply with your bank account and routing number to verify and get your refund',
                'Your account is temporarily frozen. Please log in to to secure your account ',
                #ham
                'The article was published on 18th August itself',
                'Hi George,hope ill be seeing you this evening'
    ]
```

The output of the evaluation;



```
[ ]  output

     array([['spam'],
            ['spam'],
            ['spam'],
            ['ham'],
            ['ham']], dtype='<U4')
```

As can be seen, it classified all our data correctly.

CHAPTER 5

CONCLUSION

5.1 INTRODUCTION

This chapter will outline the project's overall findings, contributions, and recommendations for improvement based on the anticipated outcomes that have been validated using the specified tools and platforms. Additionally, this project has achieved its goal, which is outlined in Chapter 1, and solved the problem.

5.2 EXPECTED RESULT

As previously said, supervised machine learning can quickly segregate messages and classify them into the appropriate categories. Additionally, it can effectively score and weigh the model. For example, the machine learning based algorithm is being used by the Gmail interface to keep spam out of users' inboxes.

5.3 LIMITATIONS AND CONSTRAINTS

According to the findings of this experiment, only text (messages), not domain names and email addresses, can be categorized and scored. This project does not attempt to prevent messages; rather, it exclusively focuses on filtering, analyzing, and classifying them.

5.4 SUGGESTIONS AND IMPROVEMENT

This project can be improved to expand its applications so that it can filter, analyze, categorize, score the model, and block spam messages in addition to text messages and other formats like domain names. These enhancements must be done if we want to classify objects with the greatest accuracy.

5.5 CONTRIBUTION

This experiment has led to the conclusion that Google Collaboratory is a cloud-based collaboration platform with the ability to predict analytics solutions for specific data. The BERT model was used in this study to identify spam.

5.6 CONCLUSION

The quality of the data source affects a classification technique's performance. In addition to lengthening the elapsed time, irrelevant and duplicated data features may also decrease the accuracy of detection. According to Chapter 2, each algorithm has distinct benefits and drawbacks.

As previously stated, supervised machine learning can efficiently segregate messages and classify them into the appropriate categories. Additionally, it is successful at scoring and weighting the model. For example, the Gmail interface uses an algorithm based on a machine learning program to keep spam out of users' inboxes.

During the implementation, only text (messages) can be classified and score instead of domain name and email address. This project only focuses on filtering, analyzing and classifying message and not blocking them. Hence, the proposed methodology may be adopted to overcome the flaws of the existing spam detection.

5.7 SUMMARY

From this project, it can be concluded that machine learning algorithm is one of the important parts in order to create spam detection application. To make it more efficient, improvement need to be implemented in future.

REFERENCES

1) Anitha, PU & Rao, Chakunta & , T.Sireesha. (2013). A Survey On: E-mail Spam Messages and Bayesian Approach for Spam Filtering. International Journal of Advanced Engineering and Global Technology (IJAEGT). 1. 124-136.

2) Attenberg, J., Weinberger, K., Dasgupta, A., Smola, A., & Zinkevich, M. (2009, July). Collaborative email-spam filtering with the hashing trick. In *Proceedings of the Sixth Conference on Email and Anti-Spam*.

3) Awad, W. A., & ELseuofi, S. M. (2011). Machine learning methods for spam e-mail classification. *International Journal of Computer Science & Information Technology (IJCSIT), 3*(1), 173-184.

4) Barnes, J. (2015). Azure Machine Learning. *Microsoft Azure Essentials. 1st ed, Microsoft*. 5) Chang, M. W., Yih, W. T., & Meek, C. (2008, August). Partitioned logistic regression for spam filtering. In *Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining* (pp. 97-105). ACM.

6) Çıltık, A., & Güngör, T. (2008). Time-efficient spam e-mail filtering using n-gram models. *Pattern Recognition Letters, 29*(1), 19-33.
48

7) Crawford, M., Khoshgoftaar, T. M., Prusa, J. D., Richter, A. N., & Al Najada, H. (2015). Survey of review spam detection using machine learning techniques. Journal of Big Data, 2(1), 23.

8) Dai, W., Xue, G. R., Yang, Q., & Yu, Y. (2007, July). Transferring naive bayes classifiers for text classification. In *AAAI* (Vol. 7, pp. 540-545).

9) Fishkin, R. (2015, November 06). Spam Score: Moz's New Metric to Measure Penalization Risk. Retrieved from https://moz.com/blog/spam-score-mozsnew-metric-to-measure-penalization-risk

10) Guzella, T. S., & Caminhas, W. M. (2009). A review of machine learning approaches to spam filtering. Expert Systems with Applications, 36(7), 10206-10222.

11) Ikonomakis, M., Kotsiantis, S., & Tampakas, V. (2005). Text classification using machine learning techniques. WSEAS transactions on computers, 4(8), 966-974.

12) Introduction | ML Universal Guides | Google Developers. (n.d.). Retrieved from https://developers.google.com/machine-learning/guides/textclassification/

14) I. Tenney, D. Das, and E. Pavlick, “Bert rediscovers the classical nlp pipeline,” *arXiv preprint arXiv:1905.05950*, 2019.

15) J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018

16) Jindal, N., & Liu, B. (2007, May). Review spam detection. In Proceedings of the 16th international conference on World Wide Web (pp. 1189-1190). ACM.

17) Joachims, T. (2002). Learning to classify text using support vector machines: Methods, theory and algorithms (Vol. 186). Norwell: Kluwer Academic Publishers.

18) Khan, A., Baharudin, B., Lee, L. H., & Khan, K. (2010). A review of machine learning algorithms for text-documents classification. *Journal of advances in*
49
information technology, *1*(1), 4-20.

19) Kolari, P., Java, A., Finin, T., Oates, T., & Joshi, A. (2006, July). Detecting spam blogs: A machine learning approach. In Proceedings of the national conference on artificial intelligence(Vol. 21, No. 2, p. 1351). Menlo Park, CA; Cambridge, MA; London; AAAI Press; MIT Press; 1999.

20) Korde, V., & Mahender, C. N. (2012). Text classification and classifiers: A survey. *International Journal of Artificial Intelligence & Applications*, *3*(2), 85.

21) Lewis, D. D., & Gale, W. A. (1994). A sequential algorithm for training text \classifiers. In *SIGIR'94* (pp. 3-12). Springer, London.

22) Lewis, D. D., & Ringuette, M. (1994, April). A comparison of two learning algorithms for text categorization. In *Third annual symposium on document analysis and information retrieval*(Vol. 33, pp. 81-93).

23) Lodhi, H., Saunders, C., Shawe-Taylor, J., Cristianini, N., & Watkins, C. (2002). Text classification using string kernels. Journal of Machine Learning Research, 2(Feb), 419-444.

24) Mahinovs, A., Tiwari, A., Roy, R., & Baxter, D. (2007). Text classification method review.

25) Rogati, M., & Yang, Y. (2002, November). High-performing feature selection for text classification. In *Proceedings of the eleventh international conference on Information and knowledge management* (pp. 659-661). ACM.

26) Sasaki, M., & Shinnou, H. (2005, November). Spam detection using text clustering.In 2005 International Conference on Cyberworlds (CW'05) (pp. 4-pp). IEEE.