INSTRUCTOR’S COMMENT:

REFEREE’S COMMENTS:

SUMMARY

Topic title: Filtering spam email using natural language processing
Student name: Le Tan Tam
Student ID: 102160192 Class: 16TCLC1
Abstract:

Nowadays, most of the people have access to the Internet, and digital world has become one of the most important parts of everybody’s life. People not only use the Internet for fun and entertainment, but also for business, banking, stock marketing, searching and so on.

Hence, the usage of the Internet is growing rapidly. One of the threats for such technology is spam. Spam is a junk mail/message or unsolicited mail/message. Spam is basically an online communication send to the user without permission. Spam has increased tremendously in the last few years.

This presentation focus on email using data from Enron Email dataset and Ling-Spam dataset. The main objective is classification spam and legit mail using pre-trained BERT model.

DA NANG UNIVERSITY
UNIVERSITY OF SICIENCE AND TECHNOLOGY
FALCUTY OF INFORMATION TECHNOLOGY

THE SOCIALIST REPUBLIC OF VIETNAM
Independence - Freedom - Happiness

GRADUATION PROJECT REQUIREMENTS

Student Name: LE TAN TAM Student ID : 102160192

Class: 16TCLC1 Faculty: Information Technology Major: Information Technology

1. *Topic title:*

   FILTERING SPAM EMAIL USING NATURAL LANGUAGE PROCESSING
2. *Project topic : ☐ has signed intellectual property agreement for final result*
3. *Initial figure and data:*

   Data is collected from Enron dataset and Ling-Spam dataset.
4. *Content of the explanations and calculations:*

   The content contains 4 parts:
   - ❖ Theories and Technologies
   - ❖ Analysis and Design
   - ❖ Text classification experiments:
     - o Data source
     - o Preprocessing data:
       - ▪ Lowercasing, punctuations and special characters removal
       - ▪ Stopwords removal
       - ▪ Stemming
       - ▪ Lemmatisation
     - o Spam email classification using BERT and visualize
   - ❖ Conclusion.
5. *Supervisor:*

   Nguyen The Xuan Ly, Information Technology Faculty, Danang University of Science and Technology, The University of Danang, Vietnam.
6. *Date of assignment : 24/08/2019*
7. *Date of completion : 04/12/2019*

Da Nang, December 04th, 2020

Head of Division…………………. **Instructor**

PREFACE

I would like to give a huge appreciation to my supervisor, Mr. NGUYEN The Xuan Ly, M.Sc. During this project, I have faced a lot of difficulties and obstacles, but Mr. NGUYEN The Xuan Ly have supported me with many advices, ideas and knowledge. Without his helps, this project could not come to life.

I also would like to express my special thanks to my family who supported, gave me motivation and help, both financially and spiritually, for this project.

And finally, I would like to say that I'm proud to be a student of Faculty of Information Technology – Danang University of Science and Technology. You have given me so much through all the years. Thanks to all the lecturers for being the coolest teachers on earth. You have taught me a lot, both inside and outside the IT professional. Most importantly, you have taught me how to think like an engineer, which I will keep carrying through my whole life. Thanks to all the IT and DUT students for being my classmates, my friends, my teammates. My youth would not be the same without you in it.

Although I tried my best to do this project, it is impossible to avoid mistakes or incompletes. I hope that I can receive the valuable comments and recommends from the teachers to complete my thesis.

Da Nang, December 2020

Students

Le Tan Tam

ASSURRANCE

I understand the University’s policy about anti-plagiarism and guarantee that:

The contents of this thesis project are performed by myself following the guidance Mr. NGUYEN The Xuan Ly.

All the references, which I used in this thesis, are quoted with author’s name, project’s name, time and location to publish clearly and faithfully.

The contents of this project is my own work and has not been copied from other sources or been previously submitted for award or assessment.

Student Performed

Le Tan Tam

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ACRONYM

- ML: Machine Learning
- NLP: Natural Language Processing
- BERT: Bidirectional Encoder Representations from Transformers
- SLM: Statistical Language Model
- KLM: Knowledge-based Language Model
- NNLM: Neural Network Language Model
- RNN: Recurrent Neural Network
- LSTM: Long Short Term Memory
- GPU: Graphics Processing Unit
- GPT-2: Generative Pre-trained Transformer 2
- NLTK: Natural Language Toolkit

INTRODUCTION

1. Reason for doing thesis:

Nowadays, most of the people have access to the Internet, and digital world has become one of the most important parts of everybody's life. People not only use the Internet for fun and entertainment, but also for business, banking, stock marketing, searching and so on.

Hence, the usage of the Internet is growing rapidly. One of the threats for such technology is spam. Spam is a junk mail/message or unsolicited mail/message. Spam is basically an online communication send to the user without permission. Spam has increased tremendously in the last few years.

Today more than 85% of mail /messages received by users are spam. These days, spam is a very serious problem because spamming has become a very profitable business for spammers. Spam email takes on various forms like adult content, selling products or services, job offers etc. Spam costs the sender very little to send but most of the costs are paid by the recipient or the service providers rather than by the sender. The cost of spam can also be measured in lost human time, lost server time and loss of valuable mail/messages.

In filtering of spam, the data cleaning of the textual information is very critical and important. Main objective of data pre-processing in spam detection is to remove data which do not give useful information about the class of the document.

2. Scope and Objective:

My project focus on email using data from Enron Email dataset and Ling-Spam dataset. The main objective is classification spam and legit mail using pre-trained BERT model.

3. Overview:

This thesis can be divided into 4 parts:

- Theories and Technology
- Analysis and Design
- Text classification experiments:
  - Data source
  - Preprocessing data:

        - Lowercasing and punctuations and special characters removal
        - Stopwords removal
        - Stemming
        - Lemmatisation
    - Spam email classification using BERT and visualize
- Conclusion.

THEORIES AND TECHNOLOGIES

1.1. Introduction:

The main focus of this thesis is about text classification, which is an application of Machine Learning and Natural Language Processing, also to compare the performance when using data pre-processing and not.

1.2. Machine Learning:

1.2.1. Definition:

In recent years, there are many achievements that human has accomplished in science and technology. These successes have made this era become the Fourth Industrial Revolution as people usually say and most of them are the implementation of AI and Machine Learning.

According to Issam El Naqa and Martin J. Murphy (2015): *"A machine learning algorithm is a computational process that uses input data to achieve a desired task without being literally programmed (i.e., "hard coded") to produce a particular outcome"*.

With this definition, we can understand that Machine Learning is a process that computers or any computational devices can learn for themselves to generate the desired output using given input.

There are many approaches in Machine Learning in term of algorithm. We can name some of the common ones:

- Supervised Learning.
- Unsupervised Learning.
- Reinforcement Learning.

1.2.2. Supervised Learning:

"*Supervised learning is the machine learning task of learning a function that maps an input to an output based on example input-output pairs*" [1].

A supervised learning algorithm analyzes the training data and produces an inferred function, which can be used for mapping new examples. An optimal scenario will allow for the algorithm to correctly determine the class labels for unseen instances. This requires the learning algorithm to generalize from the training data to unseen situations in a "reasonable" way.

Particularly, in this process, human provides the training data include the input data and the desired output of them (label). With a model of learning architecture, computer's job is learning the pattern or general rule to generate the output with the new input based on what they have learned from the old dataset.

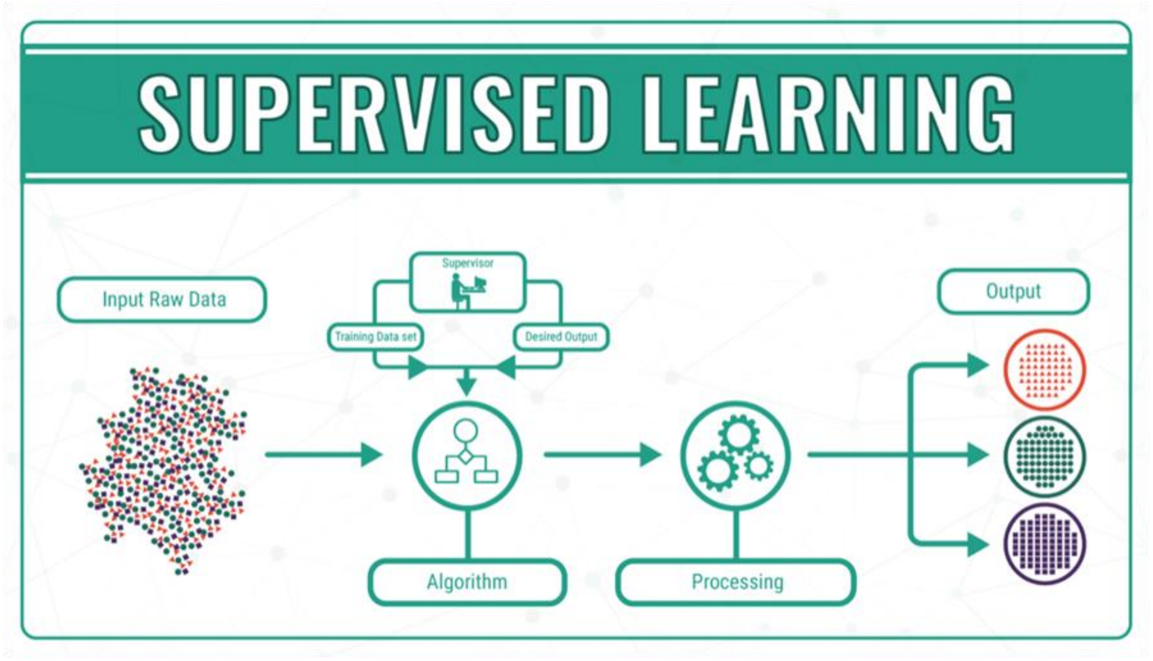


Figure 0.1: Supervised Learning

With supervised learning, you feed the output of your algorithm into the system. This means that in supervised learning, the machine already knows the output of the algorithm before it starts working on it or learning it.

With the output of the algorithm known, all that a system needs to do is to work out the steps or process needed to reach from the input to the output. The algorithm is being taught through a training data set that guides the machine. If the process goes haywire and the algorithms come up with results completely different than what should be expected, then the training data does its part to guide the algorithm back towards the right path.

Supervised Learning is widely used in practical and there are two type of application are the most popular:

- Regression: A regression model is a model that try to fit all the data points (input-output pairs) with a hyper-plane that goes through all the point (or as closest as possible). By that, we can generate a model to predict the numerical output of a new data point. Some of the common applications are:

  - Predicting the price of a house based on the variation of the price of the other houses that have some similar features.
  - Analyzing the effectiveness of marketing, promotion on sales of a product.
- Classification: These problems are related to predicting the class labels of the inputs. The program is provided with a set of inputs and the corresponding class label, its job is finding a model to predicting which class that the new input data should be in. Some of the common applications are:
  - Hand-written digits number classification.
  - Human faces recognition.

1.2.3. Unsupervised Learning:

"*This might have the objective of trying to throw a dart at a bull's-eye*".

That's what we can describe about this type of Machine Learning algorithm. In this process, computational program receives unlabeled input data to generate unpredictable outputs. Its job is finding the hidden properties of the dataset without knowing what output labels would look-a-like.

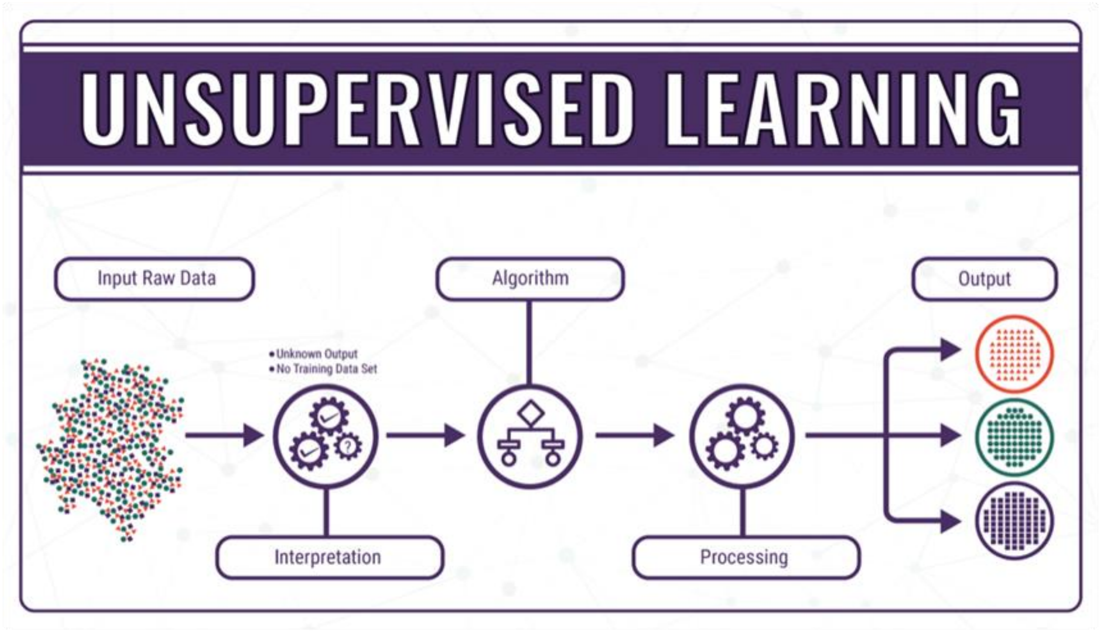


Figure 0.2: Unsupervised Learning

During the process of unsupervised learning, the system does not have concrete data sets, and the outcomes to most of the problems are largely unknown. In simple terminology, the AI system and the ML objective is blinded when it goes into the operation. The system has its faultless and immense logical operations to guide it along the way, but the lack of proper input and output algorithms makes the process even more challenging. Incredible as the whole process may sound, unsupervised learning has the ability to interpret and find solutions to a limitless amount of data, through the input data and the binary logic mechanism present in all computer systems. The system has no reference data at all.

However, in unsupervised learning, the whole process becomes a little trickier. The algorithm for an unsupervised learning system has the same input data as the one for its supervised counterpart.

Once it has the input data, the system learns all it can from the information at hand. In fact, the system works by itself to recognize the problem of classification and also the difference in shapes and colors. With information related to the problem at hand, the unsupervised learning system will then recognize all similar objects, and group them together. The labels that it will give to these objects will be designed by the machine itself. Technically, there are bound to be wrong answers, since there is a certain degree of probability. However, just like how we humans work, the strength of machine learning lies in its ability to recognize mistakes, learn from them, and to eventually make better estimations next time around.

These are some popular categories of Unsupervised Learning tasks:

- Clustering: Clustering is a process of grouping the objects, datapoints in a dataset that have some similar properties and divide them into groups. We can name a few algorithms:
    - Hierarchical clustering
    - K-means
- Anomaly detection: Detect the rare, unnormal items, datapoints that significantly different than majority of the dataset.

1.2.4. Reinforcement Learning:

Reinforcement Learning is another part of Machine Learning that is gaining a lot of prestige in how it helps the machine learn from its progress.

Reinforcement Learning spurs off from the concept of Unsupervised Learning, and gives a high sphere of control to software agents and machines to determine what the ideal behavior within a context can be. This link is formed to maximize the performance of the machine in a way that helps it to grow. Simple feedback that informs the machine about its progress is required here to help the machine learn its behavior.

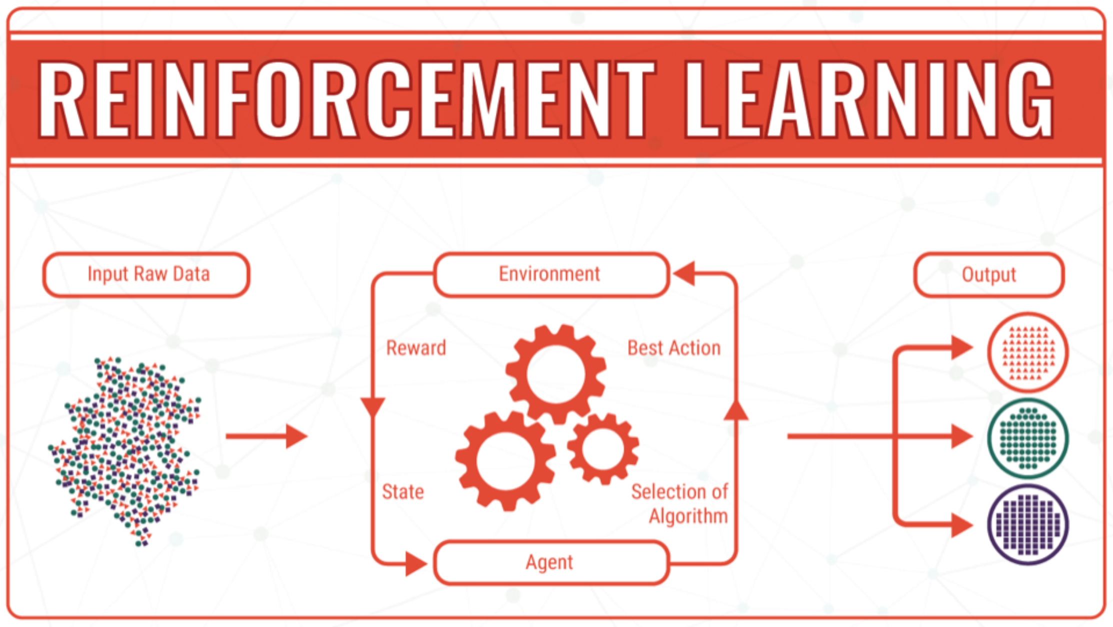


Figure 0.3: Reinforcement Learning

Reinforcement Learning is not simple, and is tackled by a plethora of different algorithms. As a matter of fact, in Reinforcement Learning an agent decides the best action based on the current state of the results.

The growth in Reinforcement Learning has led to the production of a wide variety of algorithms that help machines learn the outcome of what they are doing.

1.3. Natural Language Processing:

1.3.1. Definition:

“**Natural Language Processing** or NLP is a field of Artificial Intelligence that gives the machines the ability to read, understand and derive meaning from human languages.”

Natural language processing is a form of artificial intelligence (AI) that gives computers the ability to read, understand and interpret human language. Its mission is filling the gap between human natural language and computer understanding. It helps computers measure sentiment and determine which parts of human language are important. For computers, this is an extremely difficult thing to do because of the large amount of unstructured data, the lack of formal rules and the absence of real-world context or intent.

1.3.2. Basic tasks of Natural Language Processing:

Human language is extremely complex and diverse. That’s why natural language processing includes many techniques to interpret it, ranging from statistical and machine learning methods to rules-based and algorithmic approaches. There are five basic NLP tasks:

- **Part of speech tagging**: For every sentence, the part of speech for each word is determined. Part of speech is a category of words that have similar grammatical properties. There is a large number of words that can serve as multiple parts of speech, which makes it challenging for a machine to assign them the correct tags.
- **Lemmatization**: Lemmatization concerns removing inflectional endings only and reducing a word to its base form, which is also known as a “lemma”. Past tenses are changed into present and synonyms are unified. Lemmatization uses a different approach than stemming to reach the root form of a word.
- **Tokenization**: The tokenization task cuts a text into smaller pieces called tokens. This process segments a chunk of continuous text into separate sentences and words, while at the same time removing certain characters, like punctuation. Not all languages separate words with a blank space, therefore, in those languages, tokenization is a significant undertaking that requires deep knowledge of the vocabulary.
- **Disambiguation**: Disambiguation is a task that has to do with the meaning of the words we use in human language. Some words have more than one meaning, and while reading, we select the meaning that makes the most sense in the given context.
- **Semantics**: Humans communicate based on meaning and context. Semantics help computers identify the structure of sentences and the most relevant elements of a text in order to understand the topic that is being discussed.

1.3.3. Applications of Natural Language Processing:

There are many tasks in NLP has been developed and researched:

- **Speech Recognition**: Identify the textual representation of a given audio clip of a person speaking. This is a difficult challenge because in reality, the sound of people speaking usually is jamming by environment noises and the pause between words is not always significant to detect, not to mention the variation of the accents, dialects.

- **Natural Language Understanding**: NLP is not just about determine texts representation but also what does the texts, documents mean. There are different aspects of meaning capture like word's meaning or document's topic and many methods to do it such as word embedding or topic modeling.
- **Natural Language Generation**: This task is about transform the structured language into natural language. It also can combine with other tasks of NLP to create a more complex recognition – response system to interact with human like Alexa, Google Assistant or Siri.

1.4. Language Model:

1.4.1. Definition:

A language model is the core component of modern Natural Language Processing (NLP). It's a statistical tool that analyzes the pattern of human language for the prediction of words.

NLP-based applications use language models for a variety of tasks, such as audio to text conversion, speech recognition, sentiment analysis, summarization, spell correction, etc.

Natural language, on the other hand, isn't designed; it evolves according to the convenience and learning of an individual. There are several terms in natural language that can be used in a number of ways. This introduces ambiguity but can still be understood by humans.

Machines only understand the language of numbers. For creating language models, it is necessary to convert all the words into a sequence of numbers. For the modellers, this is known as e Encodings can be simple or complex. Generally, a number is assigned to every word and this is called label-encoding. In a sentence "I love to play cricket on weekends", every word is assigned a number [1, 2, 3, 4, 5, 6]. This is an example of how encoding is done (one-hot encoding).

1.4.2. How do Language Models work:

Language Models determine the probability of the next word by analyzing the text in data. These models interpret the data by feeding it through algorithms.

The algorithms are responsible for creating rules for the context in natural language. The models are prepared for the prediction of words by learning the features and characteristics of a language. With this learning, the model prepares itself for understanding phrases and predict the next words in sentences.

For training a language model, a number of probabilistic approaches are used. These approaches vary on the basis of purpose for which a language model is created. The amount of text data to be analyzed and the math applied for analysis makes a difference in the approach followed for creating and training a language model.

For example, a language model used for predicting the next word in a search query will be absolutely different from those used in predicting the next word in a long document (such as Google Docs). The approach followed to train the model would be unique in both cases.

1.4.3. Types of Language Models:

There are primarily two types of language models:

1.4.3.1. Statistical Language Models:

Statistical models include the development of probabilistic models that are able to predict the next word in the sequence, given the words that precede it. A number of statistical language models are in use already. Let’s take a look at some of those popular models:

- **N-Gram**: This is one of the simplest approaches to language modelling. Here, a probability distribution for a sequence of ‘n’ is created, where ‘n’ can be any number and defines the size of the gram (or sequence of words being assigned a probability). If n=4, a gram may look like: “can you help me”. Basically, ‘n’ is the amount of context that the model is trained to consider. There are different types of N-Gram models such as unigrams, bigrams, trigrams, etc.
- **Exponential**: This type of statistical model evaluates text by using an equation which is a combination of n-grams and feature functions. Here the features and parameters of the desired results are already specified. The model is based on the principle of entropy, which states that probability distribution with the most entropy is the best choice. Exponential models have fewer statistical assumptions which mean the chances of having accurate results are more.
- **Continuous Space**: In this type of statistical model, words are arranged as a non-linear combination of weights in a neural network. The process of assigning weight to a word is known as word embedding. This type of model proves

helpful in scenarios where the data set of words continues to become large and include unique words.

In cases where the data set is large and consists of rarely used or unique words, linear models such as n-gram do not work. This is because, with increasing words, the possible word sequences increase, and thus the patterns predicting the next word become weaker.

1.4.3.2. Neural Language Models

These language models are based on neural networks and are often considered as an advanced approach to execute NLP tasks. Neural language models overcome the shortcomings of classical models such as n-gram and are used for complex tasks such as speech recognition or machine translation.

Language is significantly complex and keeps on evolving. Therefore, more complex is the language model, better it would be at performing NLP tasks. Compared to the n-gram model, an exponential or continuous space model proves to be a better option for NLP tasks because they are designed to handle ambiguity and language variation.

Meanwhile, language models should be able to manage dependencies. For example, a model should be able to understand words derived from different languages.

1.4.3.3. Common Examples of Language Models

Language models are the cornerstone of Natural Language Processing (NLP) technology. We have been making the best of language models in our routine, without even realizing it. Let’s take a look at some of the examples of language models.

Speech Recognization

Voice assistants such as Siri and Alexa are examples of how language models help machines in processing speech audio.

Machine Translation

Google Translator and Microsoft Translate are examples of how NLP models can help in translating one language to another.

Sentiment Analysis

This helps in analyzing sentiments behind a phrase. This use case of NLP models is used in products that allow businesses to understand a customer’s intent behind opinions or attitudes expressed in the text. Hubspot’s Service Hub is an example of how language models can help in sentiment analysis.

Text Suggestions

Google services such as Gmail or Google Docs use language models to help users get text suggestions while they compose an email or create long text documents, respectively.

Parsing Tools

Parsing involves analyzing sentences or words that comply with syntax or grammar rules. Spell checking tools are perfect examples of language modelling and parsing.

1.5. Bidirectional Encoder Representations from Transformers (BERT):

1.5.1. Definition:

Bidirectional Encoder Representations from Transformers (BERT) is designed to pretrain deep bidirectional representations from unlabeled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be finetuned with just one additional output layer to create stateof-the-art models for a wide range of tasks, such as question answering and language inference, without substantial taskspecific architecture modifications.

BERT is conceptually simple and empirically powerful. It obtains new state-of-theart results on eleven natural language processing tasks. BERT was first created with two model size:

- BERT-BASE (L=12, H=768, A=12, Total Parameters=110M)
- BERT-LARGE (L=24, H=1024, A=16, Total Parameters=340M).

1.5.2. Transfer Learning in NLP:

Transfer learning is a technique where a deep learning model trained on a large dataset is used to perform similar tasks on another dataset. We call such a deep learning model a pre-trained model. The most renowned examples of pre-trained models are the computer vision deep learning models trained on the ImageNet dataset. So, it is better to use a pre-trained model as a starting point to solve a problem rather than building a model from scratch.

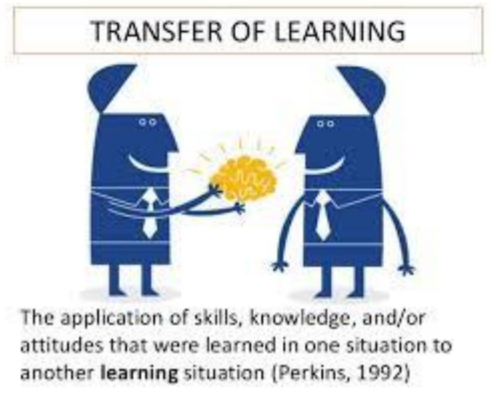

TRANSFER OF LEARNING
The application of skills, knowledge, and/or attitudes that were learned in one situation to another **learning** situation (Perkins, 1992)

Figure 0.4: Transfer of Learning

This breakthrough of transfer learning in computer vision occurred in the year 2012-13. However, with recent advances in NLP, transfer learning has become a viable option in this NLP as well.

Most of the tasks in NLP such as text classification, language modeling, machine translation, etc. are sequence modeling tasks. The traditional machine learning models and neural networks cannot capture the sequential information present in the text. Therefore, people started using recurrent neural networks (RNN and LSTM) because these architectures can model sequential information present in the text.

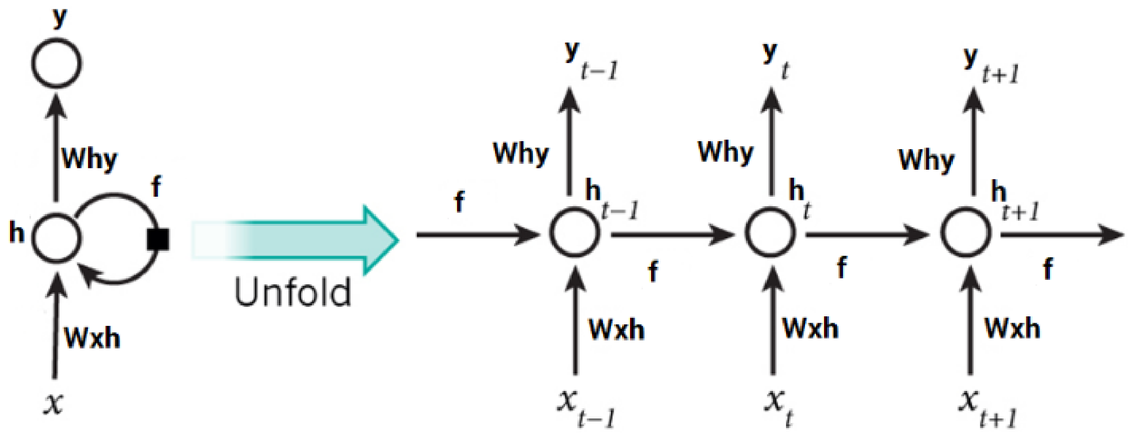


Figure 0.5: A typical RNN

However, these recurrent neural networks have their own set of problems. One major issue is that RNNs can not be parallelized because they take one input at a time. In the case of a text sequence, an RNN or LSTM would take one token at a time as input. So, it will pass through the sequence token by token. Hence, training such a model on a big dataset will take a lot of time.

So, the need for transfer learning in NLP was at an all-time high. In 2018, the transformer was introduced by Google in the paper "Attention is All You Need" which turned out to be a groundbreaking milestone in NLP.

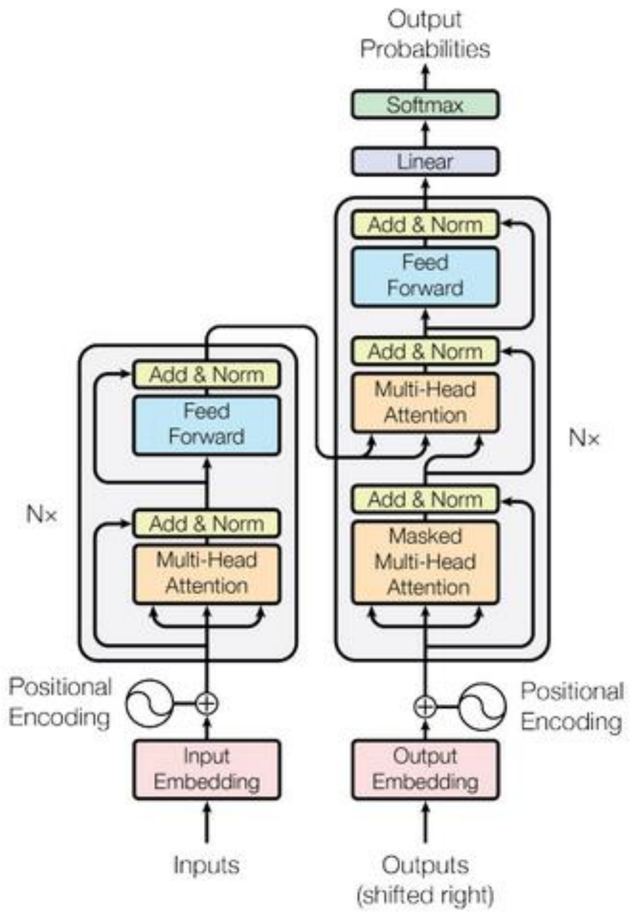


Figure 0.6: The Transformer – Model Architecture

Soon a wide range of transformer-based models started coming up for different NLP tasks. There are multiple advantages of using transformer-based models, but the most important ones are:

1st Benefit - These models do not process an input sequence token by token rather they take the entire sequence as input in one go which is a big improvement over RNN based models because now the model can be accelerated by the GPUs.

2nd Benefit - We don't need labeled data to pre-train these models. It means that we have to just provide a huge amount of unlabeled text data to train a transformer-based model. We can use this trained model for other NLP tasks like text classification, named entity recognition, text generation, etc. This is how transfer learning works in NLP.

BERT and GPT-2 are the most popular transformer-based models and in this article, we will focus on BERT and learn how we can use a pre-trained BERT model to perform text classification.

1.5.3. Input/Output Representations:

To make BERT handle a variety of down-stream tasks, BERT input representation is able to unambiguously represent both a single sentence and a pair of sentences in one token sequence. Throughout this work, a "sentence" can be an arbitrary span of contiguous text, rather than an actual linguistic sentence. A "sequence" refers to the input token sequence to BERT, which may be a single sentence or two sentences packed together. BERT use WordPiece embeddings with a 30,000 token vocabulary. The first token of every sequence is always a special classification token ([CLS]). The final hidden state corresponding to this token is used as the aggregate sequence representation for classification tasks. Sentence pairs are packed together into a single sequence. We differentiate the sentences in two ways. First, we separate them with a special token ([SEP]). Second, we add a learned embedding to every token indicating whether it belongs to sentence A or sentence B.

Input Representation:

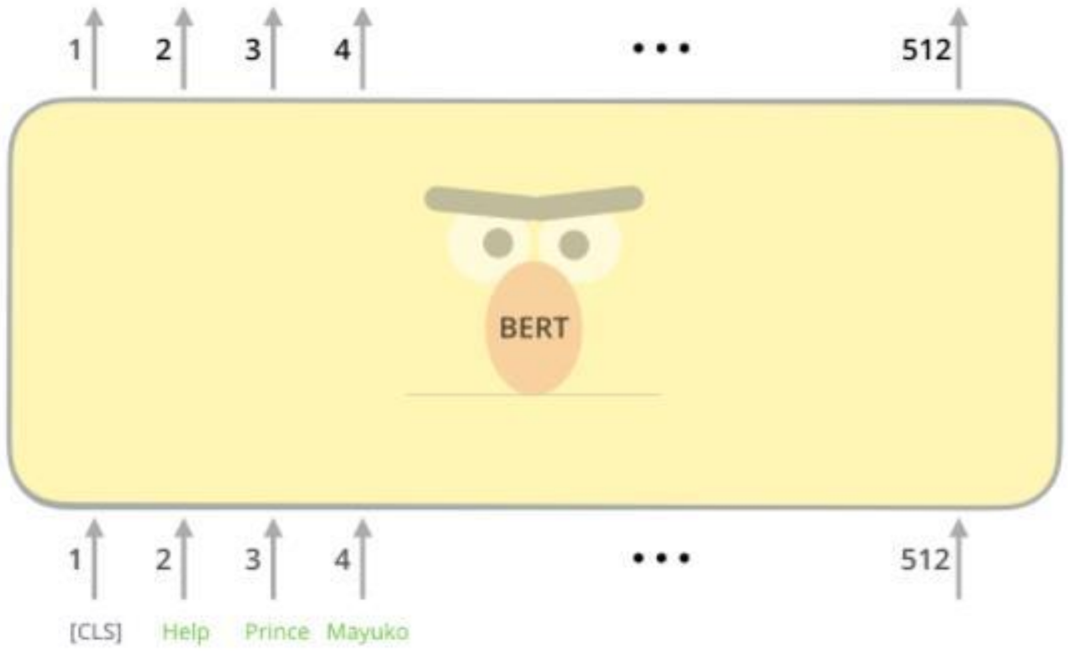


Figure 0.7: BERT input

The first input token is supplied with a special [CLS] token for reasons that will become apparent later on. CLS here stands for Classification. Just like the vanilla encoder of the transformer, BERT takes a sequence of words as input which keep flowing up the stack. Each layer applies self-attention, and passes its results through a feed-forward network, and then hands it off to the next encoder.

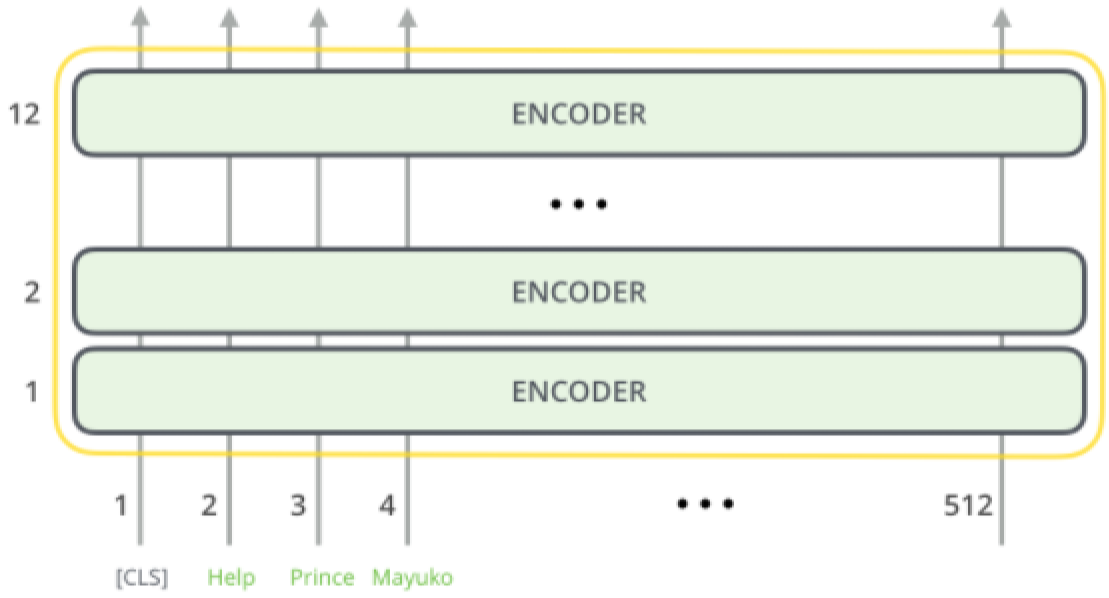


Figure 0.8: BERT input encoder layer

Output Represetation:

Each position outputs a vector of size *hidden_size* (768 in BERT Base). For the sentence classification example we've looked at above, we focus on the output of only the first position (that we passed the special [CLS] token to)

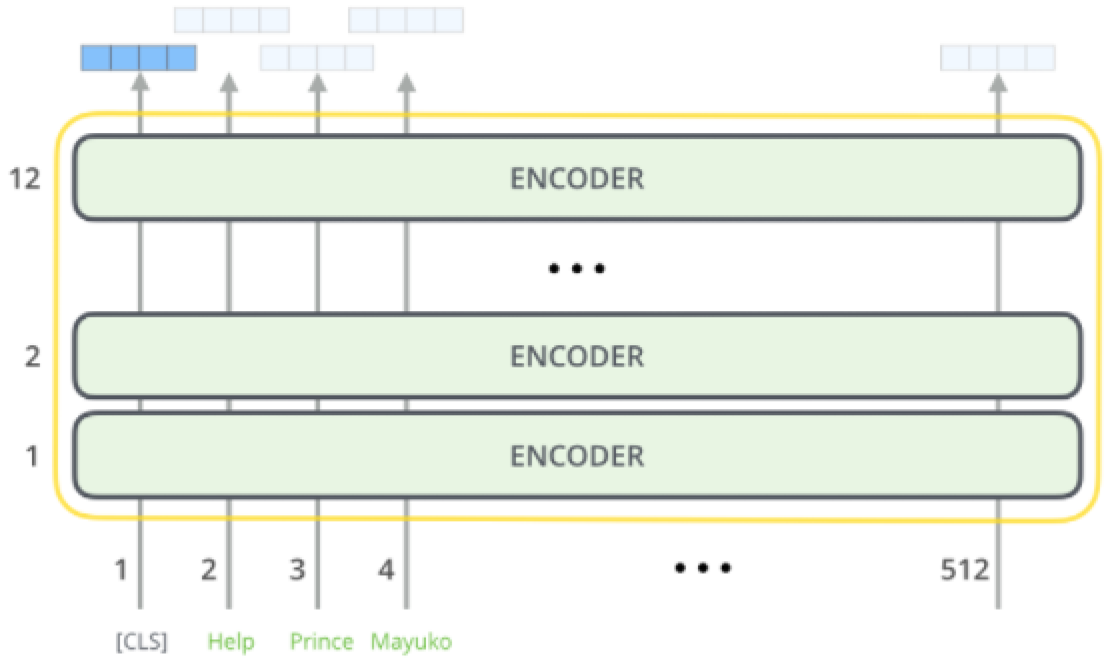


Figure 0.9: BERT output

That vector can now be used as the input for a classifier of our choosing.

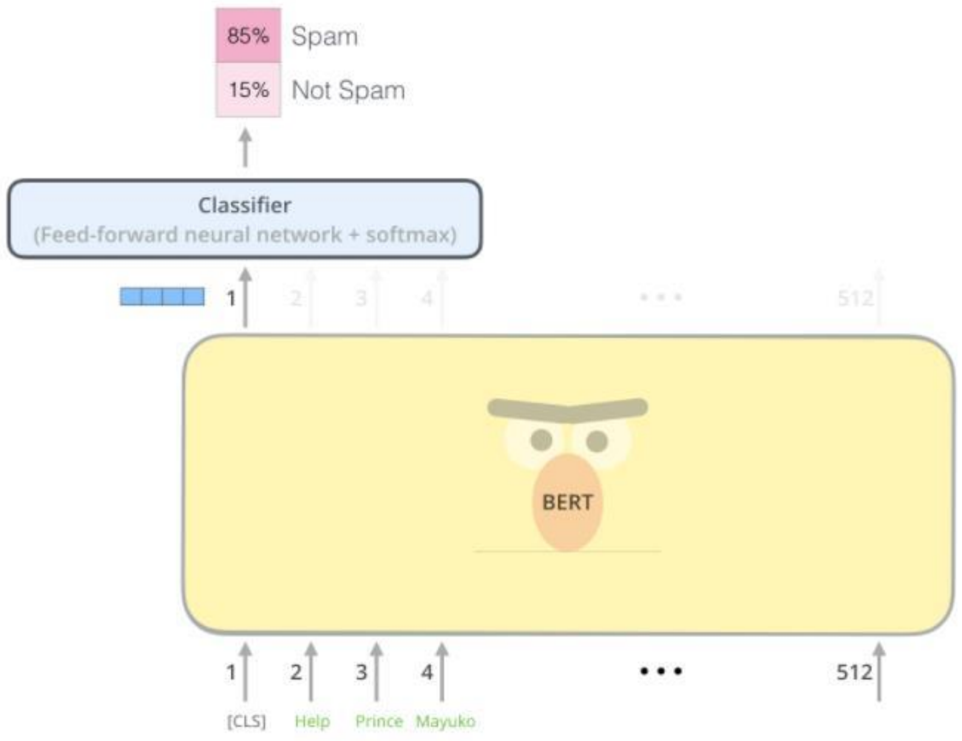


Figure 0.10: Classifier

1.5.4. How does BERT work:

There are two main steps:

- **Pre-training:** During pre-training, the model is trained on unlabeled data over different pre-training tasks.
- **Finetuning:** the BERT model is first initialized with the pre-trained parameters, and all of the parameters are fine-tuned using labeled data from the downstream tasks. Each downstream task has separate fine-tuned models, even though they are initialized with the same pre-trained parameters.

1.5.5. Pre-trained BERT:

BERT pre-training includes 2 tasks:

- **Masked Language Model:** Intuitively, it is reasonable to believe that a deep bidirectional model is strictly more powerful than either a left-to-right model or the shallow concatenation of a left-toright and a right-to-left model. Unfortunately, standard conditional language models can only be trained left-toright or right-to-left, since bidirectional conditioning would allow each word to indirectly "see itself", and the model could trivially predict the target word in a multi-layered context.

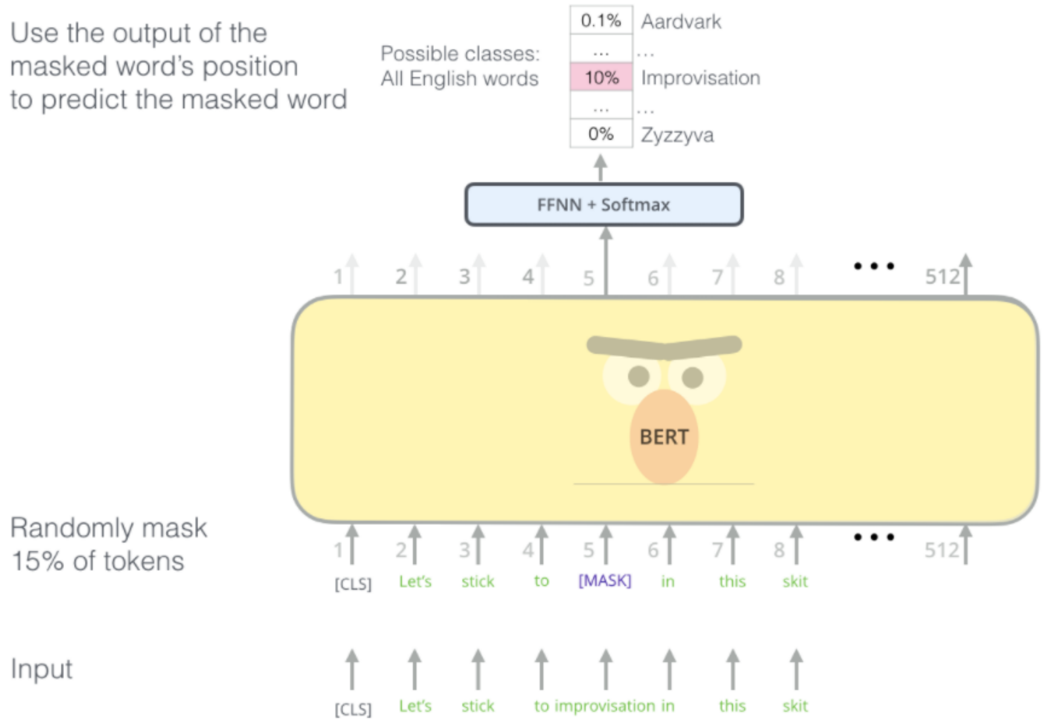

Use the output of the masked word's position to predict the masked word
Possible classes: All English words
0.1% Aardvark
...
10% Improvisation
...
0% Zyzzyva
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Randomly mask 15% of tokens
[CLS] Let's stick to [MASK] in this skit
Input
[CLS] Let's stick to improvisation in this skit

Figure 0.11: Mask Language Model

- **Next Sentence Prediction (NSP):** Many important downstream tasks such as Question Answering (QA) and Natural Language Inference (NLI) are based on understanding the relationship between two sentences, which is not directly captured by language modeling. In order to train a model that understands sentence relationships, BERT creators pre-train for a binarized next sentence prediction task that can be trivially generated from any monolingual corpus. Specifically, when choosing the sentences A and B for each pretraining example, 50% of the time B is the actual next sentence that follows A (labeled as IsNext), and 50% of the time it is a random sentence from the corpus (labeled as NotNext).

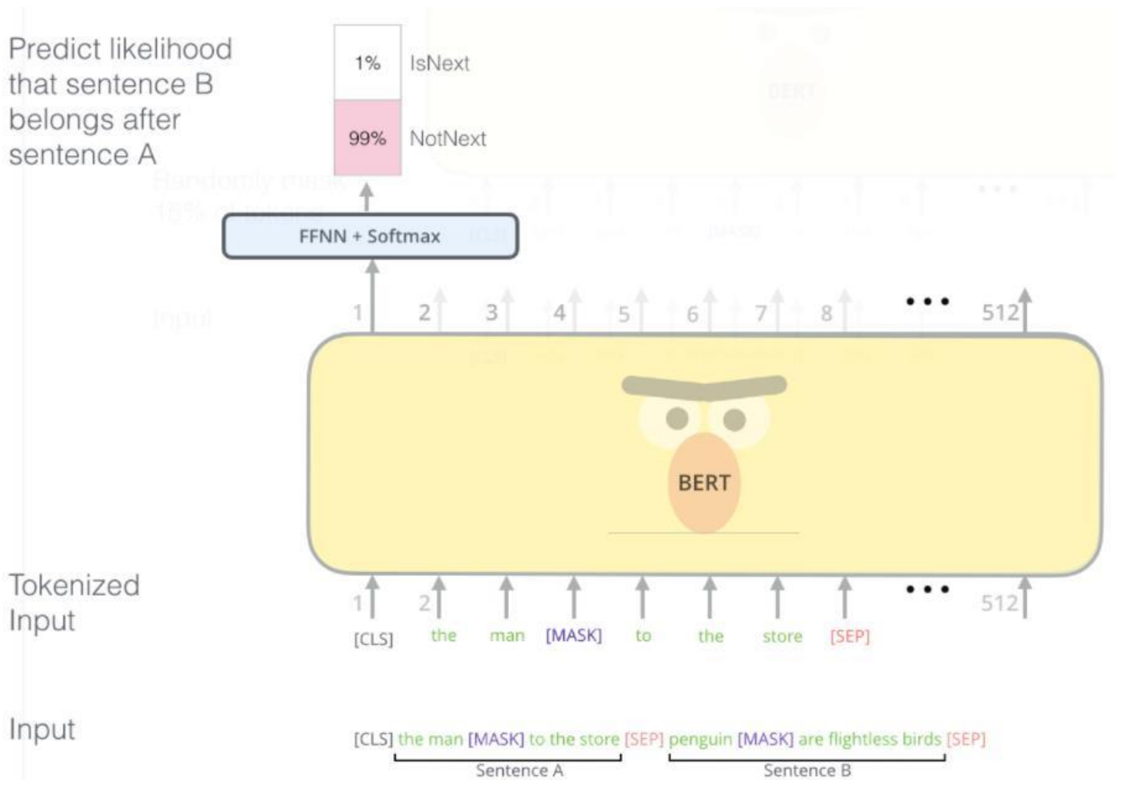

Predict likelihood that sentence B belongs after sentence A
1% IsNext
99% NotNext
FFNN + Softmax
1 2 3 4 5 6 7 8 ... 512
BERT
Tokenized Input
[CLS] the man [MASK] to the store [SEP]
Input
[CLS] the man [MASK] to the store [SEP] penguin [MASK] are flightless birds [SEP]
Sentence A
Sentence B

Figure 0.12: Next Sentence Prediction

1.5.6. Fine-tuning:

BERT (Bidirectional Encoder Representations from Transformers) is a big neural network architecture, with a huge number of parameters, that can range from 100 million to over 300 million. So, training a BERT model from scratch on a small dataset would result in overfitting.

So, it is better to use a pre-trained BERT model that was trained on a huge dataset, as a starting point. We can then further train the model on our relatively smaller dataset and this process is known as model fine-tuning.

1.5.7. Different Fine-Tuning Techniques:

Train the entire architecture – We can further train the entire pre-trained model on our dataset and feed the output to a softmax layer. In this case, the error is back-propagated through the entire architecture and the pre-trained weights of the model are updated based on the new dataset.

Train some layers while freezing others – Another way to use a pre-trained model is to train it partially. What we can do is keep the weights of initial layers of the model frozen while we retrain only the higher layers. We can try and test as to how many layers to be frozen and how many to be trained.

Freeze the entire architecture – We can even freeze all the layers of the model and attach a few neural network layers of our own and train this new model. Note that the weights of only the attached layers will be updated during model training.

In this project, I used the third approach. We will freeze all the layers of BERT during fine-tuning and append a dense layer and a softmax layer to the architecture.

ANALYSIS AND DESIGN

1.6. Data pre-processing:

1.6.1. Introduction:

Text preprocessing is traditionally an important step for natural language processing (NLP) tasks. It transforms text into a more digestible form so that machine learning algorithms can perform better.

1.6.2. Dataset:

The dataset used in the project are collected from two different sources.

- Enron Email Dataset.
- Ling-Spam Dataset.

1.6.3. Corpus statistic:

Table 0.1: Corpus statistic

Parameters	Enron Email	Ling-Spam
Duration	2004-2015	2019
Number of classes	2	2
Number of samples	91578	2893

1.6.4. Data Cleaning:

As the input data is all raw texts from news and collected from many sources by the dataset owner, all samples contain words with different tenses and plural form. The idea is to convert those words into present tense and single form, eliminate all the stop words according to the nltk stopwords corpus to make a cleaned-text field for each sample and use them for the tokenization process.

All the raw texts experience through 4 steps:

- Lowercasing and punctuations and special characters removal
- Stopwords removal.
- Stemming.
- Lemmatisation.

1.6.4.1. Lowercasing, punctuations and special characters removal:

There were many words found in the corpus that were alpha numeric. Removal of those terms was important as they do not keep on repeating in the corpus and they are just added in the emails to deceive the filter so that our classifier fails to find patterns in the given email. They do not keep on repeating in the email instances. In this sense they can be considered as unique terms. They are present in large numbers in the corpus and adding them to our features set will have drastic increase in the features set size with little of information. Counting the number of alpha numeric words in subject line or in the entire email might be helpful as spam's are reported to contain large number of alpha numeric words. So a single feature containing the number of alpha numeric words in an email might be helpful. For example:

– Input string x= " Humans only use ten percent their brains ? "

– Output string x= humans only use ten percent their brains

1.6.4.2. Removal of Stop Words

In information textual retrieval there are words that do not carry any useful information and hence are ignored during spam detection. In general and for document classification tasks we consider them as words intended to provide structure of the language rather than the content and mostly include pronouns, prepositions and conjunctions. For example:

– Input string x= humans only use ten percent their brains

– Output string x= humans ten percent brains

Table 0.2: Stop Word List for Experiment Set

then, there, that, which, the, those, now, when, which, was, were, been, had, have, has, will, subject, here, they, them, may, can, for, such, and, are, but, not, with, your, alone, anyways, along, anywhere, able, already, apart, about, also, appear, above, although, appreciate, according, always, appropriate, between, be, beyond, became, both, because brief, become, but, becomes, by, becoming, before

1.6.4.3. Stemming

The main pre-processing tasks applied in textual information retrieval tasks is the stemming. Stemming is a process of reducing words to its basic form by stripping the plural from nouns (e.g. “apples” to “apple”), the suffixes from verbs (e.g. “measuring” to “measure”) or other affixes [3]. For example, applies, applying & applied matches apply. Originally proposed by Porter on 1980, it defines stemming as a process for removing the commoner morphological and in-flexional endings from words in English [5]. In the context of document classification we can define it to be a process of representing words and its variants with its root. We used the porter stemming algorithms. For example:

– Input string x= humans ten percent brains

– Output string x= human ten percent brain

Table2 shows some examples of the words after being stemmed with porter’s algorithm

Table 0.3: Few Examples of Words with their Stems

Words	Stem
ponies	poni
caress	caress
cats	cat
feed	fe
agreed	agre
plastered	plaster
motoring	motor
sing	sing
conflated	conflat
troubling	troubl
sized	size
falling	fall

1.6.4.4. Lemmatization

In many languages, there are different words but have the same meaning also known as the synonym. Lemmatization help bring those words to only one version help decreasing the shape of the input dataset for both training and testing.

Table 0.4: Few Examples of Words Lemmatized

Words	Lemmatized
Mouse Mice Rat	Mouse
Cat Kitty	Cat

1.7. Text classification with BERT:

1.7.1. Introduction:

In this part, I will propose a solution to automatic text classification using BERT for the tokenization process.

1.7.2. Tokenization:

Tokens can be either words, characters, or subwords. Hence, tokenization can be broadly classified into 3 types – words, characters, and subwords (n-gram characters) tokenization.

- Words Tokenization
- Characters Tokenization
- Subwords Tokenization

WordPiece is the subword tokenization algorithm used for BERT (as well as DistilBERT and Electra). It relies on the same base as Byte Pair Encoding, which is to initialize the vocabulary to every character present in the corpus and progressively learn a given number of merge rules, the difference is that it doesn’t choose the pair that is the most frequent but the one that will maximize the likelihood on the corpus once merged.

Not every sample in the input data has the same length with each other and the input tensors of model fitting can not be different in length. Because of the mentioned reason, the input samples before being converted into token are added with special symbols so that every sample will be the same in length (The max length is defined by the user). Special characters to be added are:

- [CLS]: Token for text start
- [UNK]: Token for unknown word
- [SEP]: Token for text end
- [PAD]: Token for padding

1.7.3. Feature Extraction:

Not every sample in the input data has the same length with each other and the input tensors of model fitting can not be different in length. Because of the mentioned reason, the input samples before being converted into token are added with special symbols so that every sample will be the same in length (The max length is defined by the user). For BERT tokenization, [PAD] symbols along with [CLS] for sentence's start and [SEP] for sentence's end are added. In the following step, the tokenized text will be transformed into 3 vectors (Ids, Masks, Segments).

Let's take an example with the sentence "I like this project"

Table 0.5: BERT tokenized text

Tokenized text	[CLS]	I	LIKE	[UNK]	project	[SEP]	[PAD]	[PAD]
Ids	101	1234	4321	100	567	102	0	0
Masks	1	1	1	1	1	1	0	0
Segments	0	0	0	0	0	1	1	1

In this example, I chose max length of 8, as you see, 2 more [PAD] symbols were added to make the 3 vectors match the length of 8 as defined.

However, in the real project, a bigger max length should be chosen according to two reasons:

- Bigger training dataset which lead to samples with more words
- BERT splits unknown words into sub-tokens until it finds a known unigrams. For example, if a made-up word like "*zzdata*" is given, BERT would split it into ["*z*", "*##z*", "*##data*"]

We have to insert special tokens into the input text, then generate masks and segments. Finally, put all together in a tensor to get the feature matrix that will have the shape of 3 (ids, masks, segments) x Number of documents in the corpus x Sequence length. The tensor will be the training input while fitting in the model.

1.7.4. Categorical Cross-Entropy:

Categorical crossentropy is a loss function that is used in multi-class classification tasks. These are tasks where an example can only belong to one out of many possible categories, and the model must decide which one.

Formally, it is designed to quantify the difference between two probability distributions.

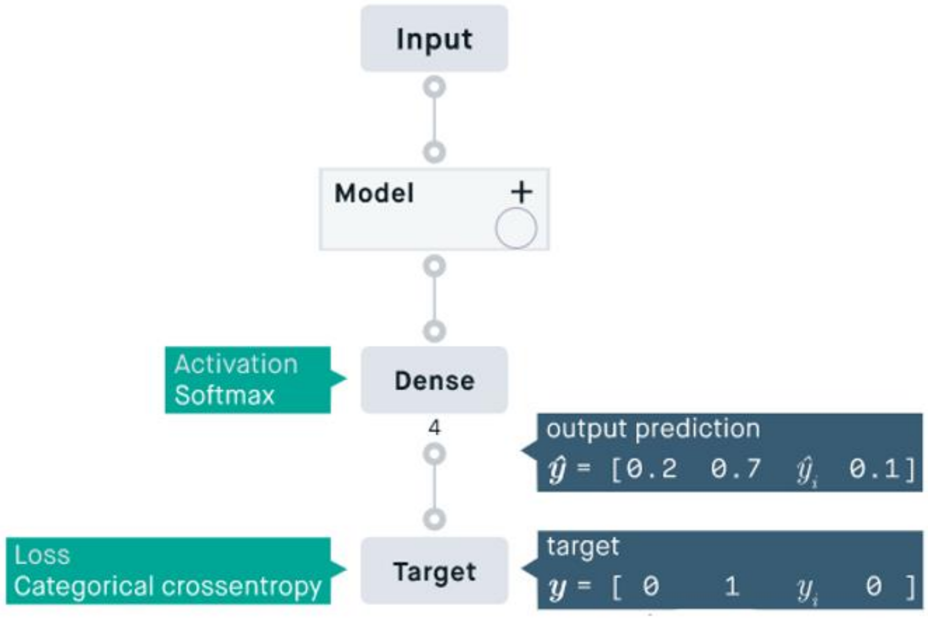


Figure 0.1: Overview of cross-entropy with activation softmax

The categorical crossentropy loss function calculates the loss of an example by computing the following sum:

$$Loss = -\sum_{i=1}^{output\ size} y_i \times log\ \hat{y}_i$$

With:

- $\hat{y}_i$: scalar value in the model output.
- y_i : the corresponding target value.
- output size: the number of scalar values in the model output.

This loss is a very good measure of how distinguishable two discrete probability distributions are from each other.

1.7.5. Softmax function:

The softmax function is used as the activation function in the output layer of neural network models that predict a multinomial probability distribution.

The function can be used as an activation function for a hidden layer in a neural network, although this is less common. It may be used when the model internally needs to choose or weight multiple different inputs at a bottleneck or concatenation layer.

By definition, the softmax activation will output one value for each node in the output layer. The output values will represent (or can be interpreted as) probabilities and the values sum to 1.0.

When modeling a multi-class classification problem, the data must be prepared. The target variable containing the class labels is first label encoded, meaning that an integer is applied to each class label from 0 to N-1, where N is the number of class labels.

The label encoded (or integer encoded) target variables are then one-hot encoded. This is a probabilistic representation of the class label, much like the softmax output. A vector is created with a position for each class label and the position. All values are marked 0 (impossible) and a 1 (certain) is used to mark the position for the class label.

For example, three class labels will be integer encoded as 0, 1, and 2. Then encoded to vectors as follows:

- Class 0:[1,0,0]
- Class 1:[0,1,0]
- Class 2:[0,0,1]

It represents the expected multinomial probability distribution for each class used to correct the model under supervised learning.

The softmax function will output a probability of class membership for each class label and attempt to best approximate the expected target for a given input.

The softmax output might look as follows, which puts the most weight on class 1 and less weight on the other classes.

The error between the expected and predicted multinomial probability distribution is often calculated using cross-entropy, and this error is then used to update the model. This is called the cross-entropy loss function.

We may want to convert the probabilities back into an integer encoded class label.

This can be achieved using the argmax() function that returns the index of the list with the largest value. Given that the class labels are integer encoded from 0 to N-1, the argmax of the probabilities will always be the integer encoded class label.

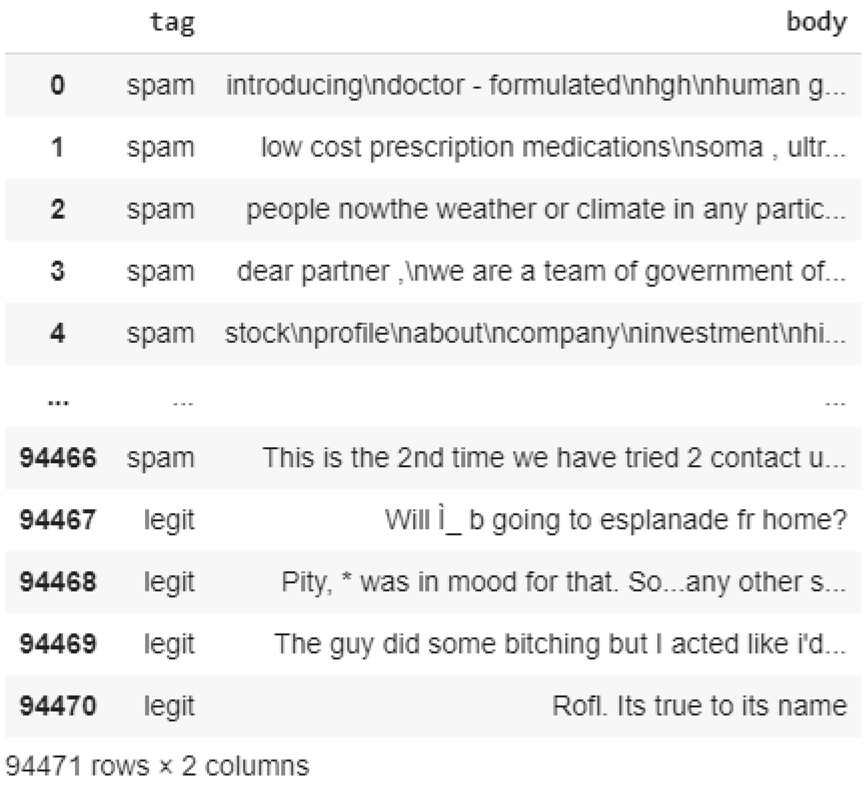
TEXT CLASSIFICATION EXPERIMENT

1.8. Processing the data source:

1.8.1. Collecting data:

The data is collected from all sources below:

- Enron Email dataset 91578 samples.
- Ling-Spam dataset with 2893 samples



	tag	body
0	spam	introducing\ndoctor - formulated\nhgh\nhuman g...
1	spam	low cost prescription medications\nsoma , ultr...
2	spam	people nowthe weather or climate in any partic...
3	spam	dear partner ,\nwe are a team of government of...
4	spam	stock\nprofile\nabout\ncompany\ninvestment\nhi...
...	...	...
94466	spam	This is the 2nd time we have tried 2 contact u...
94467	legit	Will Ì_ b going to esplanade fr home?
94468	legit	Pity, * was in mood for that. So...any other s...
94469	legit	The guy did some bitching but I acted like i'd...
94470	legit	Rofl. Its true to its name

94471 rows × 2 columns

Figure 0.1: Dataset

1.8.2. Assembling data:

For those samples with the length greater than 256 sequences, they will be cut into smaller samples with the length of 256 sequences.

The tag are the collection with the labels as text, not number, so a short script are written to transfor the text tag in to number.

After the whole process, the dataset will now has 94471 samples with the following distribution:

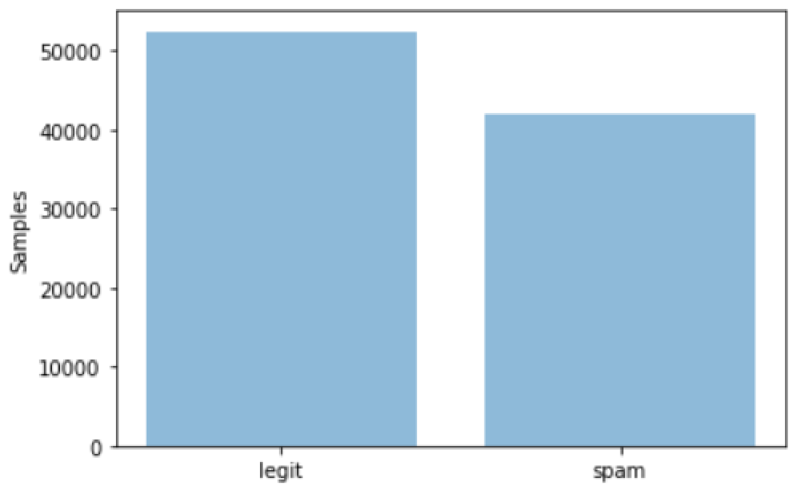


Figure 0.2: Number of samples in the dataset.

1.9. Preprocessing data:

The content of input dataset will experience through steps:

- **Step 1**: Lowercasing and punctuations and special characters removal
- **Step 2**: Remove stop words.
- **Step 3**: Stemming.
- **Step 4**: Lemmatization.

All the steps of the process will use NLTK, which is a suite of libraries for symbolic and statistical NLP for English.

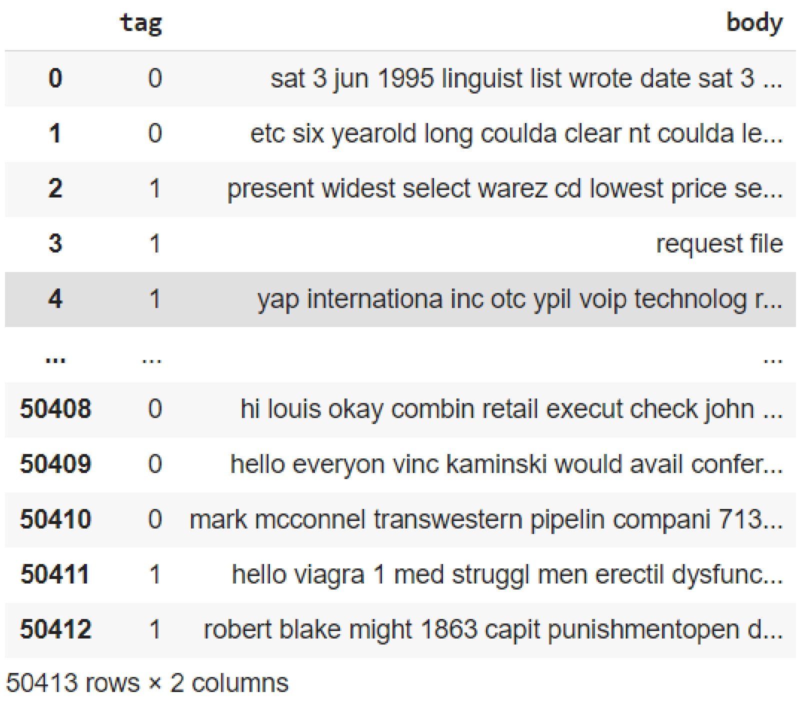
1.9.1. Natural Language Toolkit:

The Natural Language Toolkit, or more commonly NLTK, is a suite of libraries and programs for symbolic and statistical natural language processing (NLP) for English written in the Python programming language. It was developed by Steven Bird and Edward Loper in the Department of Computer and Information Science at the University of Pennsylvania. NLTK includes graphical demonstrations and sample data. It is accompanied by a book that explains the underlying concepts behind the language processing tasks supported by the toolkit, plus a cookbook.

NLTK is intended to support research and teaching in NLP or closely related areas, including empirical linguistics, cognitive science, artificial intelligence, information retrieval, and machine learning. NLTK has been used successfully as a teaching tool, as an individual study tool, and as a platform for prototyping and building research systems. There are 32 universities in the US and 25 countries using NLTK in their courses. NLTK supports classification, tokenization, stemming, tagging, parsing, and semantic reasoning functionalities.

Figure 0.3: The Natural Language Toolkit in Python

1.9.2. Process result:



	tag	body
0	0	sat 3 jun 1995 linguist list wrote date sat 3 ...
1	0	etc six yearold long coulda clear nt coulda le...
2	1	present widest select warez cd lowest price se...
3	1	request file
4	1	yap internationa inc otc ypil voip technolog r...
...	...	...
50408	0	hi louis okay combin retail execut check john ...
50409	0	hello everyon vinc kaminski would avail confer...
50410	0	mark mcconnel transwestern pipelin compani 713...
50411	1	hello viagra 1 med struggl men erectil dysfunc...
50412	1	robert blake might 1863 capit punishmentopen d...

50413 rows × 2 columns

Figure 0.4: Cleaned dataset

1.9.3. Tokenization:

I tried to encode sentences using the BertTokenizerFast.

Table 0.1: Encoded sentences using tokenizer

raw sentence	introduc doctor formul hgh human growth hormon also call hgh refer medic scienc master hormon plenti young near age twenti one bodi begin produc less time forti nearli everyon defici hgh eighti product normal diminish least 90 95 advantag hgh increas muscl strength loss bodi fat increas bone densiti lower blood pressur quicken wound heal reduc cellulit improv vision wrinkl disappear increas skin thick textur increas energi level improv sleep emot stabil improv memori mental alert increas sexual potenc resist common ill strengthen heart muscl control cholesterol control mood swing new hair growth color restor read websit unsubscrib
encoded sentence	[101, 2830, 26445, 16480, 27488, 13058, 4372, 4948, 5757, 5757, 2456, 2184, 2459, 3889, 28925, 5757, 5757, 2456, 2184, 2385, 5388, 10507, 3395, 15868, 5187, 2549, 7065, 2483, 3466, 1020, 1021, 4002, 12072, 9447, 1052, 1048, 22512, 2156, 22476, 11132, 2225, 7232, 5187, 2549, 3466, 1020, 1021, 4002, 4067, 3889, 15868, 5187, 2549, 7065, 12879, 14876, 3438, 19841, 2692, 28712, 102, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0, 0]

1.10. BERT:

1.10.1. Feature Extraction:

For BERT approach, I will do the transfer learning from two pre-trained model of BERT, BERT-Uncased with 110 millions parameters and Distil-BERT, a lighter, faster training version of BERT with 66 millions parameters thanks to the remove of pooler and token – type embeddings (segments). I use BERT-base tokenizer from pretrained BERT.

After declaring the tokenizer, I select the sequence max length of 256 because BERT splits unknown words into sub-tokens until it finds a known unigrams.

The next steps is generating three embeddings (ids, masks and segments) from the input training dataset. The summary of the three embeddings is the Feature matrix and also the training input

1.10.2. Define Model Architecture:

- **Step 1:** Freeze all the layers of the model before fine-tuning it.

This will prevent updating of model weights during fine-tuning. If you wish to fine-tune even the pre-trained weights of the BERT model then you should not execute the code above.

- **Step 2:** Define my model architecture. Pass the pre-trained BERT to our define architecture
- **Step 3:** Use AdamW as our optimizer. It is an improved version of the Adam optimizer.
- **Step 4:** Compute class weights, pass these weights to the loss function

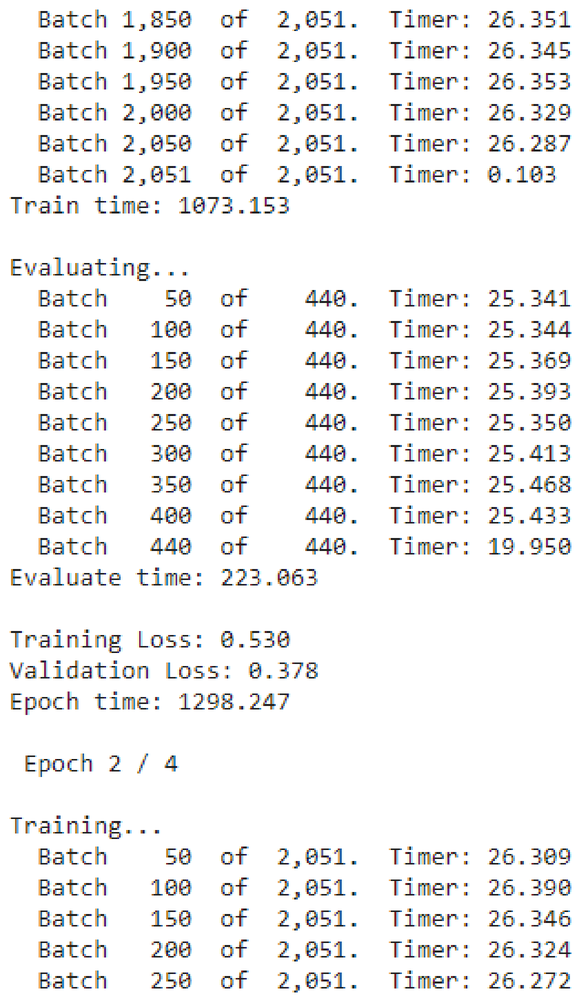
There is a class imbalance in dataset. The majority of the observations are not spam. So, I will first compute class weights for the labels in the train set and then pass these weights to the loss function so that it takes care of the class imbalance.

1.10.3. Fine-tuning BERT:

Now I have to define a couple of functions to train (fine-tune) and evaluate the model, respectively.

DataLoader package used to read batch size of data to mamory in the case dataset is larger than memory.

After each epoch I saved the best model weights for future use.

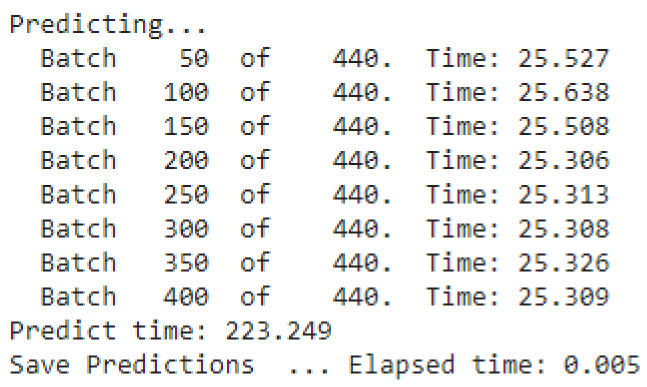


```
  Batch 1,850  of  2,051.  Timer: 26.351
  Batch 1,900  of  2,051.  Timer: 26.345
  Batch 1,950  of  2,051.  Timer: 26.353
  Batch 2,000  of  2,051.  Timer: 26.329
  Batch 2,050  of  2,051.  Timer: 26.287
  Batch 2,051  of  2,051.  Timer: 0.103
Train time: 1073.153

Evaluating...
  Batch    50  of    440.  Timer: 25.341
  Batch   100  of    440.  Timer: 25.344
  Batch   150  of    440.  Timer: 25.369
  Batch   200  of    440.  Timer: 25.393
  Batch   250  of    440.  Timer: 25.350
  Batch   300  of    440.  Timer: 25.413
  Batch   350  of    440.  Timer: 25.468
  Batch   400  of    440.  Timer: 25.433
  Batch   440  of    440.  Timer: 19.950
Evaluate time: 223.063

Training Loss: 0.530
Validation Loss: 0.378
Epoch time: 1298.247

 Epoch 2 / 4

Training...
  Batch    50  of  2,051.  Timer: 26.309
  Batch   100  of  2,051.  Timer: 26.390
  Batch   150  of  2,051.  Timer: 26.346
  Batch   200  of  2,051.  Timer: 26.324
  Batch   250  of  2,051.  Timer: 26.272
```

Figure 0.5: Training model

1.10.4. Get Predictions for Test data:

Fisrt of all, load the best model weights which were saved during the training process. Once the weights are loaded, I can use the fine-tuned model to make predictions on the test set. I saved the predictions for future use.

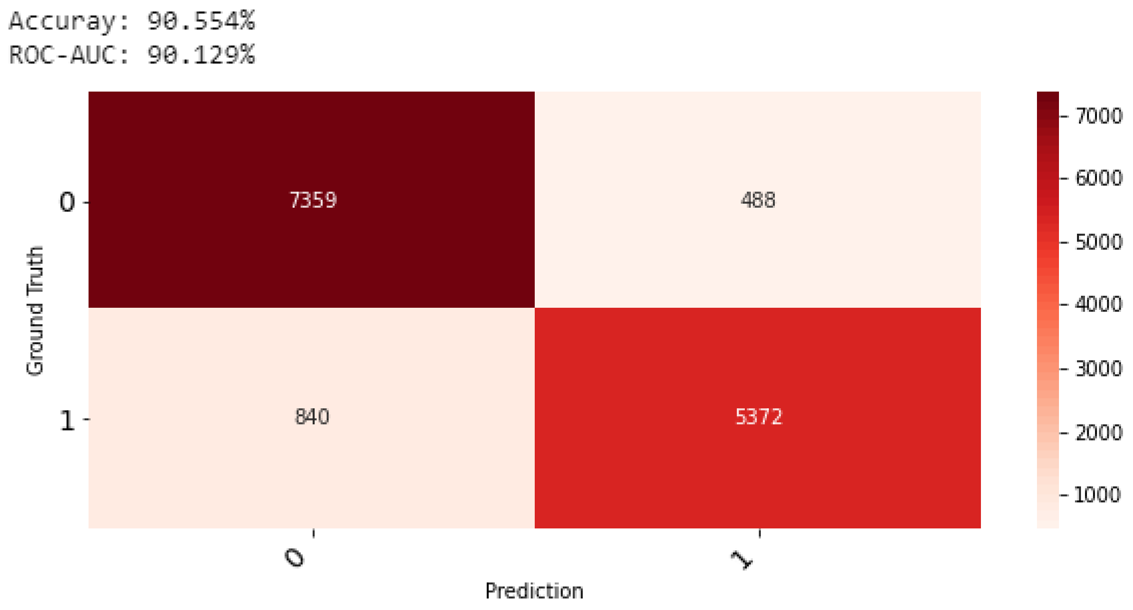


```
Predicting...
  Batch    50  of    440.  Time: 25.527
  Batch   100  of    440.  Time: 25.638
  Batch   150  of    440.  Time: 25.508
  Batch   200  of    440.  Time: 25.306
  Batch   250  of    440.  Time: 25.313
  Batch   300  of    440.  Time: 25.308
  Batch   350  of    440.  Time: 25.326
  Batch   400  of    440.  Time: 25.309
Predict time: 223.249
Save Predictions  ... Elapsed time: 0.005
```

Figure 0.6: Predicting

I trained two models, one using pre-processed data, the other not. Below are the accuracy of two models with test data:

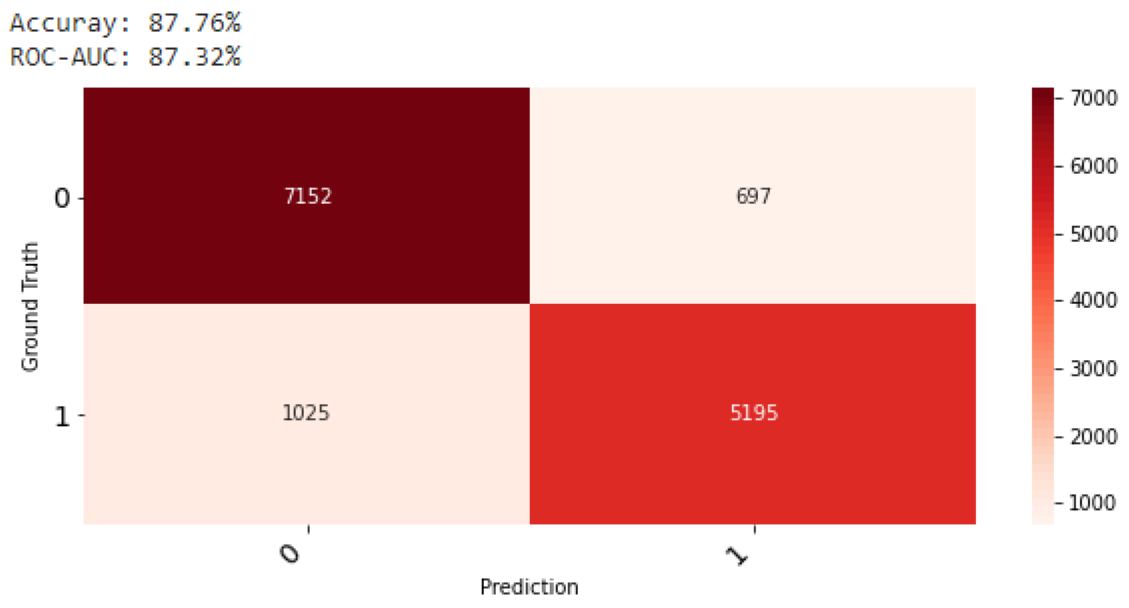
Pre-processed dataset:



```
Accuray: 90.554%
ROC-AUC: 90.129%
```

Figure 0.7: Accuracy of pre-processed data

Raw dataset:



```
Accuray: 87.76%
ROC-AUC: 87.32%
```

Figure 0.8: Accuracy of raw data

CONCLUSION

1. Conclusion:

In the process of our senior project, I have learned the basic and needed knowledge to prepare for the job, especially the knowledge of machine learning, natural language processing and Python that supplemented the missing information and many experiences in the work process. Learning new technologies and apply it to a project that enhances my knowledge and practical experience. In addition, during the Senior project help me a lot on learn how to divide work, manage time, manage workflow and familiarize with the work environment and pressure, working on time and response the rules.

2.Development orientation:

To be more widely used in real life and as a tool for people to clean their mailbox in the future, I will research more methods of increasing accuracy and find better dataset.

REFERENCES

[1] Wikipedia **Supervised Learning:** https://en.wikipedia.org/wiki/Supervised_learning

[2] Supervised Learning in *Ronald van Loon - Machine Learning Explained: Understanding Supervised, Unsupervised and Reinforcement Learning.*

[3] Unsupervised Learning in *Ronald van Loon - Machine Learning Explained: Understanding Supervised, Unsupervised and Reinforcement Learning.*

[4] Reinforcement Learning in *Ronald van Loon - Machine Learning Explained: Understanding Supervised, Unsupervised and Reinforcement Learning.*

[5] How machine process and understand human language in *Diego Lopez Yse – Your Guide to Natural Language Processing (NLP).*

[6] What is Natural Language Processing in *Kyrill Poelmans – What is natural language processing (NLP)?*

[7] Five basic NLP tasks in *Kyrill Poelmans – What is natural language processing (NLP)?*

[8] Wikipedia **Natural language processing**: https://en.wikipedia.org/wiki/Natural_language_processing

[9] Wikipedia **Language Model**: https://en.wikipedia.org/wiki/Language_model

[10] Universal Language Model Fine-tuning *in Jeremy Howard, Sebastian Ruder - Universal Language Model Fine-tuning for Text Classification.*

[11] Wikipedia **BERT:** https://en.wikipedia.org/wiki/BERT_(language_model)

[12] BERT in *Jacob Devlin, Ming-Wei Chang, Kenton Lee, Kristina Toutanova BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

[13] BERT-transformers: https://huggingface.co/transformers/model_doc/bert.html

[14] Enron Email Dataset : https://www.cs.cmu.edu/~./enron/

[15] Ling-Spam Dataset : https://www.kaggle.com/mandygu/lingspam-dataset\

[16] Jacob Devlin, Ming-Wei Chang, Kenton Lee and Kristina Toutanova (2019). In: *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding.*

[17] Categorical cross-entropy: https://peltarion.com/knowledge-center/documentation/modeling-view/build-an-ai-model/loss-functions/categorical-crossentropy

[18] Softmax activation with Python https://machinelearningmastery.com/softmax-activation-function-with-python/

[19] Categorical cross-entropy in *Peltarion, Building an AI, Loss functions.*

[20] Softmax activation in *Jason Brownlee* - *Softmax Activation with Python.*

[21] Wikipedia **Natural Language Toolkit:** https://github.com/nltk/nltk/wiki