Tunisian Republic

Ministry of Higher Education
and Scientific Research

University of Carthage

National Institute of Applied Science and Technologies

PERSONAL PROFESSIONAL PROJECT REPORT

Computer Networking and Telecommunications Engineer's Degree

Implemented and Presented By

Oumayma Rjab

Oussama Jaouadi

Taha Mediouni

Deep Learning For Email Classification

Academic supervisor: **Dr. Sofiane Ouni** Assistant professor

University Year 2020 - 2021

Résumé

De nos jours, le moyen de communication professionnel le plus utilisés est l'email d'où la quantité énorme d'email reçut chaque jour. Cela nécessite le développement d'un système automatisé de categorisation des emails. En raison des ressources limitées aucune extension pour la classification des emails n'a abordé à ce jour la détection automatisée des catégories dépendant du contenu textuel de l'email. Des opérations de traitement du langage naturel (NLP) sont effectuées afin d'examiner et de catégoriser ces emails. Ce projet est une tâche de détection de catégorie en utilisant les techniques les plus récentes en deep learning et en faisant l'étude du modele du Google le plus récente BERT.

Mots clés : ENRON Email Dataset,Catégorisation des emails, NLP, BERT, Deep Learning, extension de navigateur.

Abstract

Nowadays, the most used means of professional communication is the email from which the huge amount of email received each day. This requires the development of an automated email categorization system. Due to limited resources no extension for the classification of emails has so far addressed the automated detection of categories depending on the textual content of the email. Natural Language Processing (NLP) operations are performed to review and categorize these emails. This project is a category detection task using the latest deep learning techniques and studying the Google model of the latest BERT.

Keywords : ENRON Email Dataset, Email Categorization, NLP, BERT, Deep Learning, Browser Extension.

Dedication

I dedicate this work to:

, **Dr. Sofiane Ouni** for mentoring us and doing everything they could to assist us.

Oumayma Rjab

Oussama Jaouadi & Taha Mediouni

Table of Contents

List of Figures

List of Tables

Abbreviations list

- **BERT** = Bidirectional Encoder Representations from Transformers
- **NLP** = Natural Language Pre-Processing
- **BiLSTM** = Bidirectional Long short-term memory
- **RNN** = Recurrent neural network
- **RoBERTa** = Robustly Optimized BERT Pretraining Approach
- **BOW** = Bag of words
- **ELMO** = Embeddings from Language Models
- **SVM** = Support vector machine
- **MLM** = Masked language model
- **RR** = Reciprical-Rank
- **API** = Application Programming Interface

General Introduction

Email is a popular mode of communication in today's society for a variety of reasons. Email can be delivered instantly, it is great for preserving records since it creates a virtual trail of who sent it, what it was about, and what it contained, it is a good way to promote items, and it is the most cost-effective way for businesses to communicate. It's typical to have many email addresses these days: personal emails, work emails, and school / college emails. Every day, most people receive between 20 and 40 emails. For some, it may be a lot more. Going through each and every email to see if it is of any importance is a time-consuming chore. Many of the emails you receive are promotional offers from stores or periodicals to which you have subscribed. It has the potential to overflow the inbox, resulting in a disorganized mess. Though recent progress has been made by email providers, by allowing the creation of an email classification system for sorting/grouping emails.

In this digital age of 'smart computing', it is logical to have an automatic task for organizing emails, a classification algorithm to process the email and organize them into appropriate classes in accordance with user relevance.

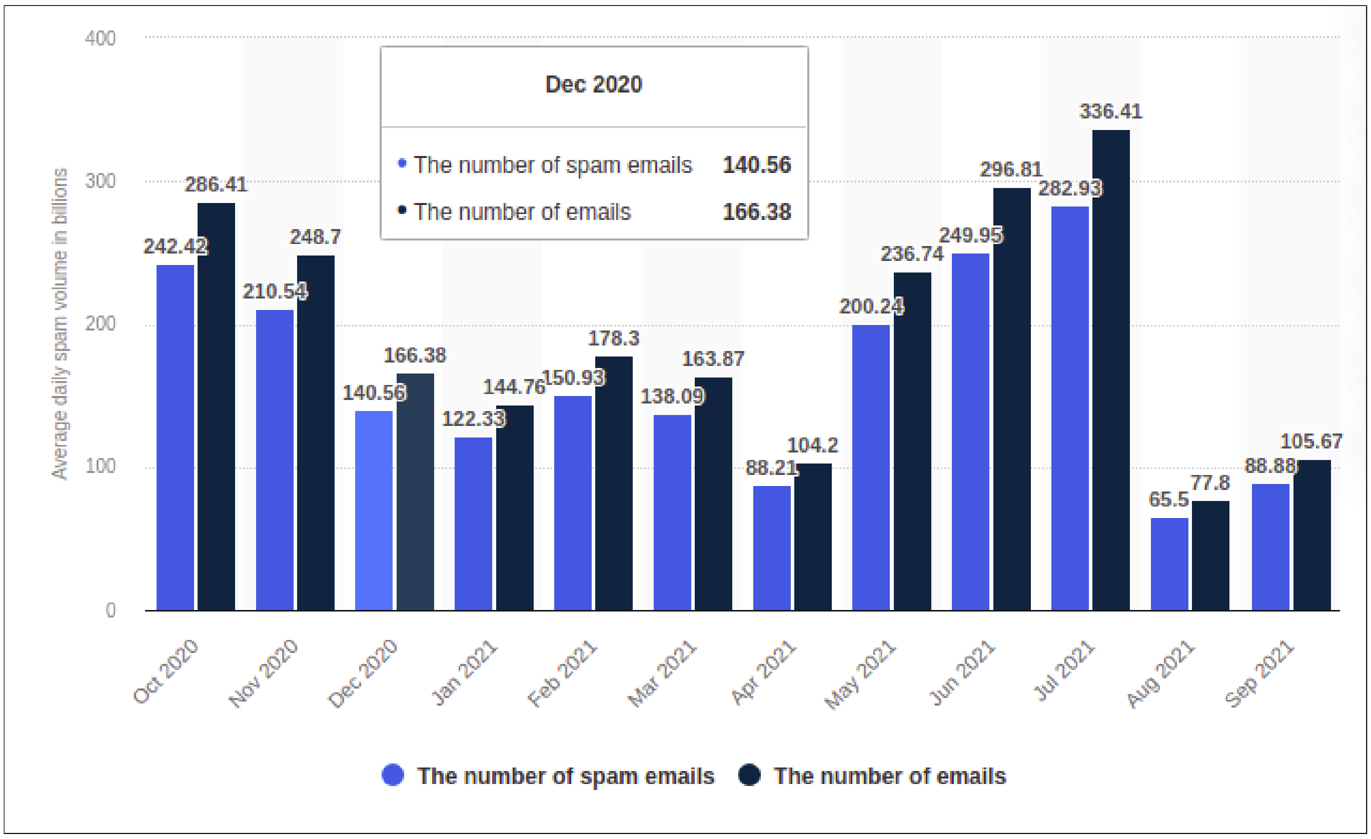


Figure 0.1: Number of spam emails received

This report is arranged as follows: The first chapter describes the hosting company as the final part of the study project development process, the structure of the project topic, the specific problems we address, the purpose of the study, the methodology for the research project including information on the selection of methods and strategies used in the research, approach and limitations of the study and the contributions of the study to the state. The second chapter covers the current literature and conceptual foundation with which the research is interrelated. The third chapter provides approaches and strategies in the methodology employed. Then the fourth chapter discusses the deep learning approach and model results. The fifth and the last chapter discusses the extension development. Finally, we close our report with an overall conclusion summarizing our findings.

Chapter 1

General context

Plan

Introduction

This chapter presents the general context of the project. We are going to introduce the work environment, the problem statement, and the goal behind this study. Furthermore, we are going to announce the delimitations of the project and its contributions.

1.1 Problem Statement

With the number of Internet users increasing daily, email has become the most extensively used communication mechanism. It is estimated that there are more than 4 billion email accounts all over the world. They are estimated to reach 4.6 billion by 2025. Even kids are allowed under certain conditions to have emails under the supervision of their parents.

In recent years, the increased use of email has led to the appearance of many problems caused by spam and phishing emails. A typical email user receives 40 to 50 emails per day and for others hundred emails per day are considered as usual. They spend a significant part of their working time on processing emails. Spam in emails is considered as the most complex problem in email services. Spam emails are those unwanted, unsolicited emails that are not intended for a specific receiver and that are sent for either marketing purposes, or for scam, hoaxes, etc. It is estimated that nearly 85 percent of all emails are classified as spam.

Therefore, email classification has become a serious issue to individuals and organizations and it necessitates the need to create a mechanism that intelligently classifies and deals with the problem.

Importance of Email Classification

When we receive an email, we are receiving data and with a seamlessly integrated classification ribbon in the toolbar at the top of the page, it becomes simple to classify the data, differentiating the significant from the insignificant.

With an email classification system, any email containing unsolicited content will be automatically flagged as spam. It also saves users' time and increases efficiency. It assists in highlighting emails and knowing their contents without spending hours scrolling through the email box.

Because users receive so many emails per day, any opportunity to reduce noise helps reduce the time that a user spends processing his emails.

1.2 Purpose of the Study

This dissertation's main goals start with the creation of a system that is able to classify the new emails for a user. This classification must be significantly correct and fast, reducing user frustration and augmenting productivity. Second, the system needs to be easily integrated with other applications and consume as less resources as possible. Finally, the resulting system must be successfully validated.

1.3 Approach and Boundaries of the Study

1.3.1 Research method used

As that the final output will be a building Web Browser extension that classify emails,but also during this project we will study new Deep learning approaches that's why we will adheres to the design science methodology. This project will deliver and study a new model and labeled dataset in the field of Natural Language Processing, especially in email classification.

1.3.2 Scope

The scope of this study is to develop an understanding of an email by different Natural Language Processing techniques and Deep Learning language models in order to improve the Accuracy of the classification model.

1.3.3 Limitations

The main issue is that emails fall into the category of unstructured data that is typically text heavy. This results in irregularities and ambiguities that make it difficult to automate the process of analyzing and categorizing the emails.

1.4 Contributions

An email classification system is a monitoring system from within an email inbox, designed to monitor the real-time flow of email into an organization.
It also gives users the ability to classify emails in accordance with their sensitivity. This ensures that they have better control over data.

Conclusion

Although email is a faster, better alternative to paper documents and faxing, it can have a direct impact on corporate bottom lines by distracting workers from role-relevant tasks to spend hours daily dealing with unimportant messages.
This project will contribute to research by serving as a jumping-off point for the NLP community interested in this topic. In the next chapter, we will describe the key notion of NLP and the packages required, as well as the current research in this assignment.

Chapter 2

Background and Literature review

Plan

Introduction

E-mail classification has been a popular study area in recent years and since theoretical knowledge is important before diving into the code we are going to define some important concepts starting with the NLP itself and e-mail classification approaches and existing studies.

2.1 Background

Natural Language Processing (NLP)

Natural Language Processing, also known as NLP, is a branch of Artificial Intelligence that enables machines to interpret, understand, and extract meaningful information from human languages. NLP will assist you with a variety of activities, and the domains of application seem to be growing on a daily basis. Machine Translation, Speech Recognition, Named Entity Recognition, Automatic Summarization, Chat-bots, Market Intelligence, Text Classification, and other NLP applications are only a few examples. This project is part of the NLP field's Text Classification application, specifically for email classification and spam detection.

Email Classification Definition

An email classification system is a monitoring system that monitors the real-time flow of emails into an organization from within an email inbox. The system classifies each email depending on the parameters provided by the individual or enterprise by automatically evaluating the content of the email body and any attachments. For example, in Gmail, Primary, Social, Promotions, Updates, and Forums are some of the tabs used to organize emails. The system classifies emails into the preferred folder by extracting the email sender and content, as well as learning user behavior. Instead of depending on yourself to manually classify emails as potential inducements, an email classification system automates 99% of the process and integrates smoothly into business and compliance operations.

Neural Networks

Neural networks are made up of layers that are formed of neurons and the connections which link them. A layer is a bunch of neurons that receive input and produce output. In the training process, the hyper-parameters of the layers are initialized randomly and then revised making use of the back-propagation theorem Back propagation is used to minimize function loss Applying mathematical equations, the loss function provides the model's error rate. Activation functions are mathematical calculations that define a neural network's output. A feed-forward neural network is one in which the connections between nodes do not form a cycle. 3 types of layers are common to all neural networks:

Input layer: contains raw data

Hidden layer: where the process of learning is applied

Output layer: contains the output predictions

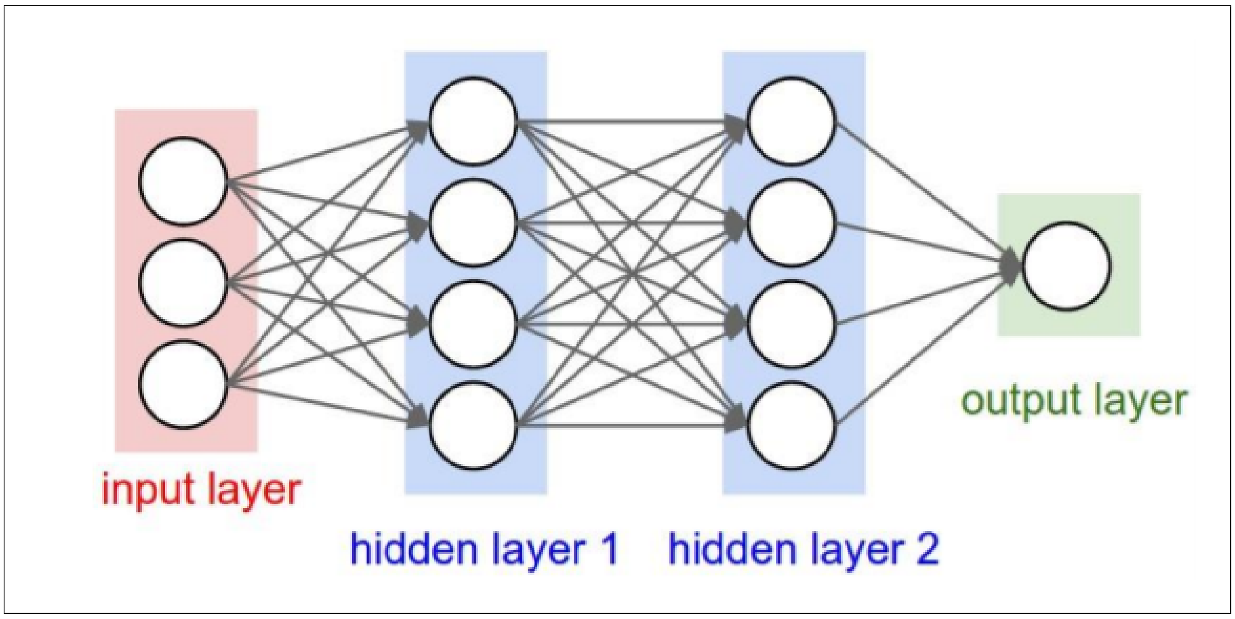


Figure 2.1: Simple Neural Network

Recurrent Neural Networks(RNN)

Recurrent Neural Networks or RNN (Quast and The Graduate Institute 2016) are a form of Artificial Neural Network which is commonly seen in text classification due to their sequential tendency that facilitates the capture of meaning in texts.

Recursion encourages branching in hierarchical feature spaces, and as training progresses, the eventual network architecture resembles the process.

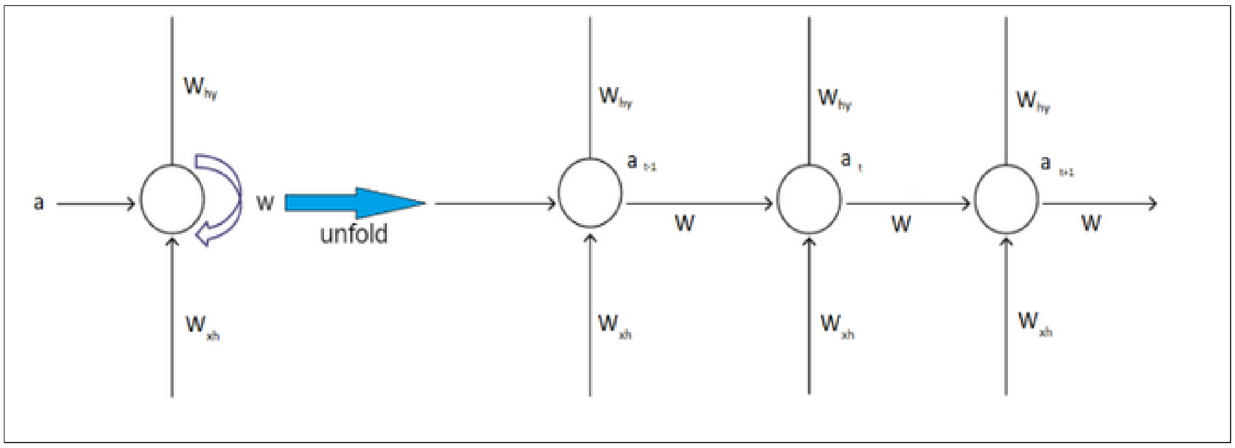


Figure 2.2: RNN architecture

Bidirectional Long-Short Term Memory

RNN represents a type of Neural network that is widely used for text classification problems because of its sequential architecture. Each neuron in the RNN is updated based on the weights and biases of the previous neuron.

One type of recurrent neural network is the Long Short-Term Memory (LSTM). The Long-Short Term Memory only considers the historical information of the sequence, but it is often not enough. If you could access future information as you would access the previous one, this is would be very beneficial for the sequence of tasks.

The bidirectional LSTM (Rhanoui et al., n.d.) addresses this problem. Bi LSTM consists of a forward Long-Short Term Memory layer and a backward Long-Short Term Memory layer. More contextual information can be extracted using a bidirectional LSTM [1].

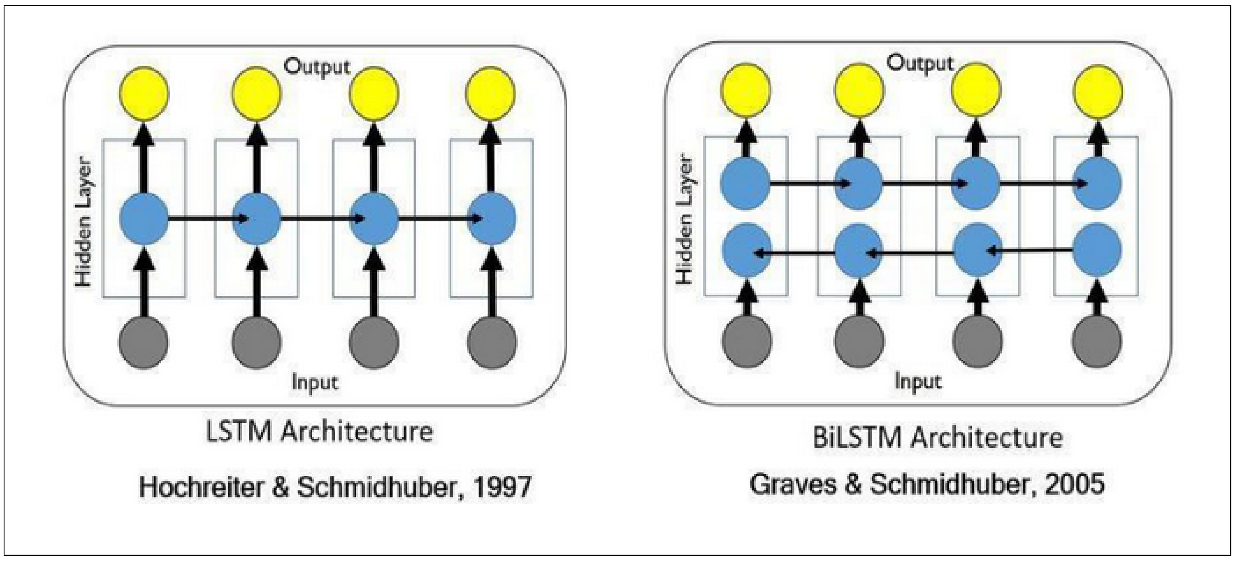


Figure 2.3: LSTM vs Bi-LSTM architecture

Transfer Learning

Transfer learning is a methodology that uses a deep learning model that has been trained on a large dataset to accomplish similar tasks on some other dataset. A deep learning model such as this is referred to as a pre-trained model. As a result, it is preferable to use pre-trained models rather than developing a model from scratch. So most NLP tasks were at an all-time high of need for transfer learning since RNN are sequential and they process the inputs token by token and it will take a lot of time on a large dataset [2].

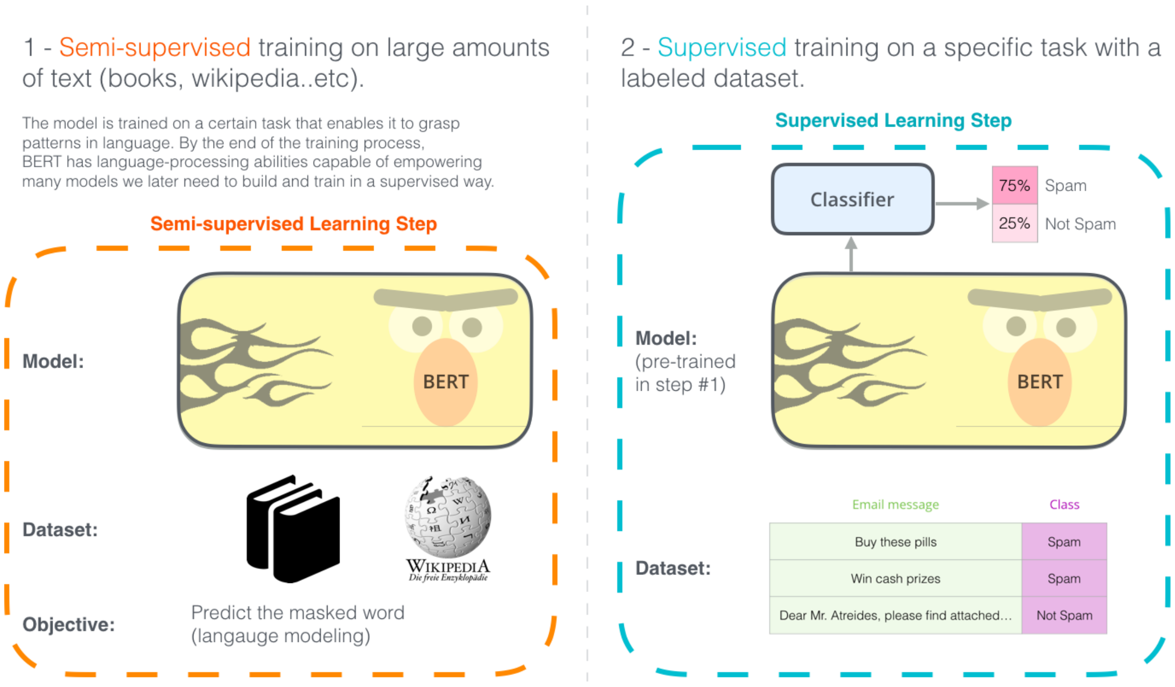
Self-Attention

Attention [3] is basically a matrix that represents word relativity. Self-Attention is the focusing of one's attention on oneself. The Self-Attention Layer checks attention of all terms in the same sentence at the same time, making it a basic matrix calculation that can be parallelized through computing units.

Deep Transformers

Transformer [4] uses stacked self-attention fully connected layers to build blocks of encoders and decoders. Some transformer-based models include BERT, which employs two stages of training: first, they use transformers to train a natural language model without labels to initialize the parameters, and then they train transformers with labels to adapt the parameters to a new supervised task.

Bidirectional Encoder Representations from Transformers (BERT)


1 - Semi-supervised training on large amounts of text (books, wikipedia..etc).
The model is trained on a certain task that enables it to grasp patterns in language. By the end of the training process, BERT has language-processing abilities capable of empowering many models we later need to build and train in a supervised way.
Semi-supervised Learning Step
Model:
BERT
Dataset:
WIKIPEDIA
Die freie Enzyklopädie
Objective:
Predict the masked word (langauge modeling)
2 - Supervised training on a specific task with a labeled dataset.
Supervised Learning Step
Classifier
75% Spam
25% Not Spam
Model:
(pre-trained in step #1)
BERT
Dataset:

Email message	Class
Buy these pills	Spam
Win cash prizes	Spam
Dear Mr. Atreides, please find attached…	Not Spam

Figure 2.4: Fine tuning BERT [5]

- **Model Architecture**

BERT [6] is a trained Transformer Encoder stack. The BERT model sizes have many encoder layers (also called Transformer Blocks) – twelve in the Base version. Also, it has wider feed forward networks (768 hidden units) and 12 attention heads.

- **Model Input**

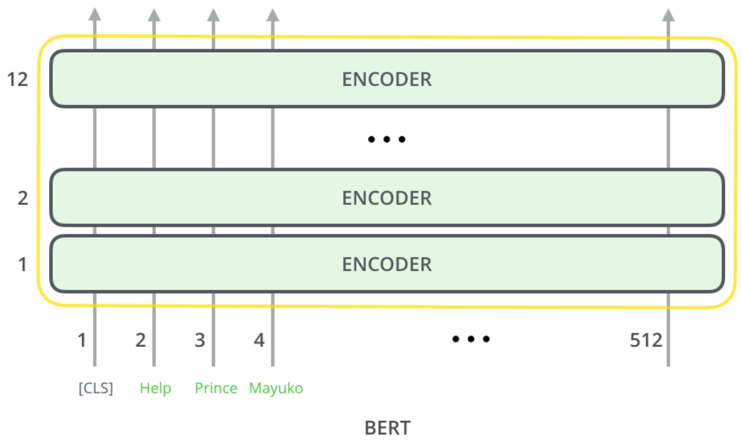


Figure 2.5: BERT Input Example [5]

A special [CLS] token is provided with the first input token. CLS stands for Classification in this case. BERT receives a series of terms as data, which continues to flow up the stack. Each encoder performs self-attention, passes the results through a feed-forward network, and then passes the information on to the next encoder.

- **Model Output**

Each position produces a hidden size vector (768 in BERT Base). We concentrate on performance of only the first position in the sentence (that we passed the special [CLS] token to).
This vector will now be used as the input for our classification. By using a single-layer neural network as a classifier, we can obtain great performance.

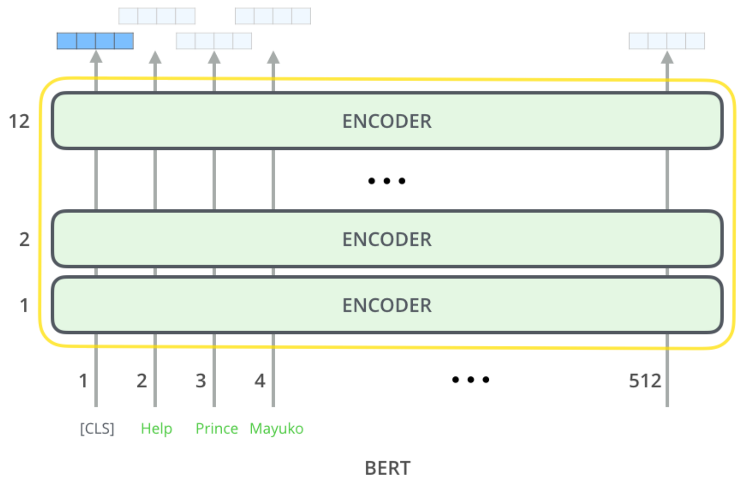


Figure 2.6: BERT Output Example [5]

- **RoBERTa**

The self-supervised transformers model RoBERTa was trained on a vast corpus of English data. This means it was pre-trained on solely raw texts with no human annotation (which explains why it can use so much publicly available data), and then utilized an automated way to generate inputs and labels from those texts.

It had been pre-trained with the goal of Masked language modeling (MLM) in mind. Using a sentence as an example, the model randomly masks 15% of the words in the input, then runs the entire masked text through the model, which must predict the masked words. In contrast to traditional recurrent neural networks (RNNs), which generally perceive words one after the other, or auto regressive models like GPT, which internally hide future tokens, this model sees words one after the other. It allows the model to learn the bidirectional representation of a sentence.

In this way, the model learns an inner representation of the English language, which can then be used to extract features relevant for downstream tasks: for example, if you have a dataset of labeled sentences, the features learned by the BERT model can be used as inputs to train a typical classifier [7].

Word Embddings

Word Embeddings is a vector representation of words into vectors of real numbers. Different techniques exist:

- CountVectorizer : it counts the number of words in the document i.e it converts a collection of text documents to a matrix of the counts of occurrences of each word in the document.

- TF-IDF (stands for Term-Frequency-Inverse-Document Frequency) weights down the common words occuring in almost all the documents and give more importance to the words that appear in a subset of documents. TFIDF works by penalising these common words by assigning them lower weights while giving importance to some rare words in a particular document.

2.2 Existing Related Studies

A paper under the title of "Manipulation of Email Data Using Machine Learning and Data Visualization" presents spam and non-spam recognition using some classical machine learning algorithms (decision tree, naive Bayes, support vector machine....). (Verma Sinha, n.d.) The approach was a simple generation of dataset, Training data set, dictionary creation, and finally Implementation of the classification algorithm. The data they used was a sum of 1257 emails with 999 of them being non-spam emails and 258 were spam emails. The data was split into 62% for training and the rest for testing.

- The first step was making the email dataset from the Gmail data for each message denote the phases associated with the f=f1.....f2 then define a function that maps the set of fields F into features.
- The second step was building feature vectors: extracting features from the subject and the body then extracting features from the sender and the subdomain.

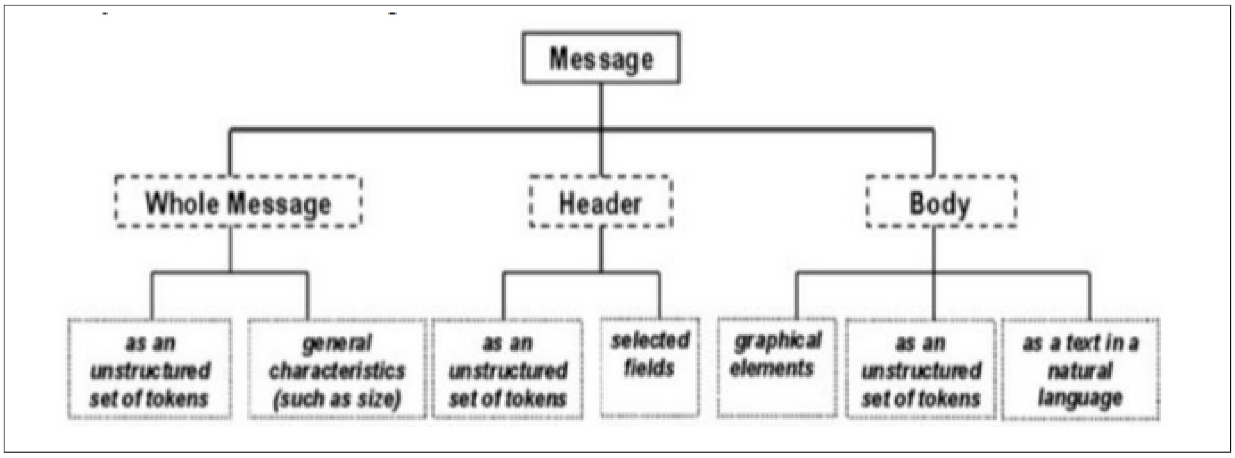

Message
Whole Message
Header
Body
as an unstructured set of tokens
general characteristics (such as size)
as an unstructured set of tokens
selected fields
graphical elements
as an unstructured set of tokens
as a text in a natural language

Figure 2.7: In structuring the message

Then they have to choose their models and compare them. The models were :

1. Rule based classifier :

Rule based classifier follows the concept of if-then rules for text classification. They are mutually exclusive and exhaustive in nature; areas of applications of rule-based classifiers are Biology spam detection and filtering.

2. Decision tree :

Decision tree follows “divide and conquer” or “rule of classification algorithm”. tree consists of the attributes and the emerging branches, close to the top-down approach skating the root node and implementing the set of rules, and selecting the branch with the nearest neighbor. Decision trees are widely used in the field of computational biology and Biology Informatics.

3. Naive Bayes :

It follows the probabilistic classifying algorithm based on the best theorem with independent assumptions.

4. Support vector machines :

It's a machine-learning algorithm that employs the supervised learning algorithm. If it is given a set of labeled data training sets then the support vector machines Model out hyperplanes that classify the set and assign them to the appropriate group.

5. k-nearest neighbor :

Classifies the documents using K nearest neighbor follow up the similarity measure like in the distance measure. Nearest neighbor is a kind of machine learning algorithm in which the classification is done on the labeled training data set and choosing the best out of the minimum and smallest distance from the object.

- The final step was to evaluate our model to choose the best :

We define a metric or accuracy as number of well classified / number of total classifications . As a metric, they have used a confusing matrix which is a specific table layout that allows visualization of the performance of an algorithm. Each row of the matrix represents the instances in an actual class while each column represents the instances in a predicted class, or vice versa – both variants are found in the literature. There were 970 emails non-spam and well classified but 49 spam emails were wrongly classified also there were 209 spam emails well classified and 29 spam emails were classified as non-spam. After many algorithms, the authors stated that there is a pattern or a set of words in spam emails and non-spam emails so they extracted them and plotted them in the following figure. To conclude the paper used many simple machine learning algorithms into a dataset after extracting features from email and the algorithms were used, tested, and evaluated with a confusion matrix finally we were able to extract relevant words repeating in spam mail.

Another paper under the title of "Email Classification into Relevant Category Using Neural Networks" (Gupta, Goyal, and Reckon Analytics 2018,) presents two experiments for email categorization. The main issue to starting this study was that several online purchasing sites or service providers provide a single email address to which clients may submit their questions, concerns, and so on. So how can a back-end service provider determine which emails belong to which departments when they get millions of emails per week?

This research proposed a model of an artificial neural network (ANN) to tackle the challenge of email

classifica tion The dataset to execute all experiments in this study, authors have used data from personal Gmail inboxes. The dataset contains 608 emails and a total word count of 42219 that have been labeled with the predefined Gmail label function that labeled them in 9 classes.
To define the model and since Keras (Chollet, F., and others 2015) provides a playground for simple and quick implementation of deep learning experiments in Python, the authors chose to develop neural networks using keras in this study.
In addition, Keras converts emails into a numeric matrix by assigning a rank based on the number of times a word appears, and then turning the number into vectors that reflect each email's label. To appropriately categorize fresh emails, a neural network was trained by consuming this data with LSTM. However, neural network accuracy improves by identifying the best and worst features for each email using the Keras tokenizer.
Before starting the dataset was splitted in 548 emails used to train the model, and 60 emails used to validate it.
Two experiments were run to evaluate the model's performance for various input parameters.

- In the first experiment, the authors fine tune the model based on the number of hidden layers with a fixed number of words. At first they fixed only 1 hidden layer in the model architecture to achieve 33% accuracy. Then as they augmented the number of hidden layers the accuracy increased until they achieved stability in performance with 85% accuracy at the use of 100 hidden layers.

- In the second experiment, the researchers applied pre-processing techniques to the data, such as deleting English verbs and conjunction words, assuming that these words would not enhance accuracy. They also fixed the number of hidden layers up to 100 and fine-tuned the number of words in order to increase the model's performance, which they achieved by raising the number of words by 118% and boosting the accuracy to 88.33%.

In conclusion from the above study, more words per email enhance accuracy, but classes with a small number of emails do not process to achieve precision, which is why a large dataset is essential for this email classification task.

Conclusion

There are a lot of related studies in this field, but none for a customized solution, so this chapter defined the key concepts used in this project. Studying others' work and the state of the art results will help us better understand the problem and achieve good results on our dataset and pipeline. Before we move to code, we must define the methodology and how the work is going to be organized.

Chapter 3

Methodology

Plan

Introduction

This section will address the methodology shown in this project. It describes the data and its design and properties. It also covers how data preprocessing and analyzing were performed. As well as Deep Learning approach and experiments, BERT model experiments, ML model deployment and extension development.

3.1 Design Science methodology

In all of its iterations and phases, this project adheres to Design Science methodology. Figure 3.1 shows the five steps of each iteration. The investigation of the problem is the first step of an iteration. The researcher recognizes the issue statement, its effect on the business flow, its additions to the domain, and so on throughout this goal-oriented process. Then, based on the literature analysis, the collected related work a solution design is proposed. Following that, potential solutions are validated with team members along with the criteria established during the problem investigation process. The chosen solutions are implemented during the development process or implementation phase which can be represented as developing a tool, implementing code or running a model. Finally, an evaluation is made to see if the issue under investigation has been resolved. At the completion of each iteration, the next iteration's input is the current one output's.

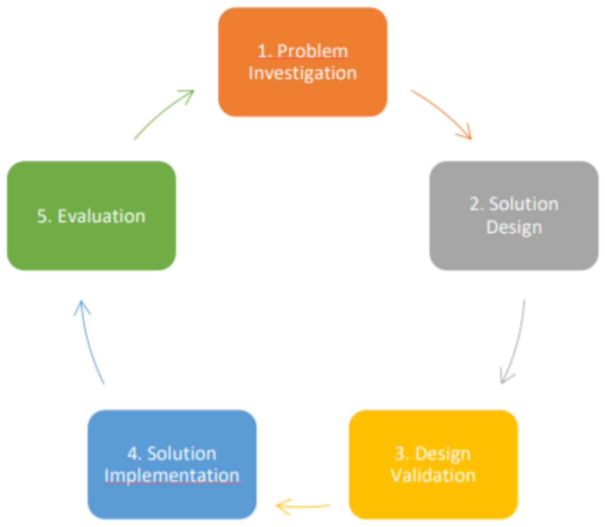


Figure 3.1: Design Science methodology phases of one iteration

3.2 Iteration 1: Data Preprocessing & Analyzing

Email cleaning is defined as a process of eliminating irrelevant non-text data (which includes header, signature, forwarded message and program code).

So before classification, an email goes through multiple pre-processing procedures that remove useless data and improve the overall quality of the information provided in the email. The email headers and the message body are extracted. Then, the emails are cleaned and transformed which effectively reduces the email into a single line string.

3.3 Iteration 2 : Deep Learning Approach & Experiments

The Long Short Term Memory network (LSTM) is an enhanced RNN (sequential network) that permits information to be stored indefinitely. It can deal with the vanishing gradient problem that RNN has. For persistent memory, a recurrent neural network, also known as RNN, is used.

In this section, an LSTM was implemented and Google's pre-trained Word Embeddings were used and after a bit of fine-tuning, the best parameters were found.

3.3.1 Word Embedding

Word Embedding is a process that converts text to numerical that involves representing characters, words, or sentences as a sequence of vectors in a high-dimensional space. Word Embedding is used to generate vectors that will be input into the model. in this section, BERT embeddings are used so the model's inputs is a phrase instead of a word because BERT creates contextual embeddings and requires knowledge of the context or adjacent words before producing a vector representation. unlike other embeddings models such as Doc2Vec that doesn't take in consideration the word position.

3.3.2 Evaluation

The models are evaluated by the following metrics :

Accuracy: the accuracy of a model refers to how similar a predicted value is to a True or known value using the formula:

Accuracy = (TP + TN)/ (TP + TN + FP + FN) where :

TP represents True Positive: positive predicted values are true

FP represents False Positive: negative predicted values are true

FN represents False Negative: positive predicted values are false

TN represents True Negative: negative predicted values are false

3.4 Iteration 3 : BERT Model Experiments

BERT is an acronym for Bidirectional Encoder Representations from Transformers. Essentially, it's a Transformers alteration in which the encoder is kept and the decoder is discarded. It achieved state-of-the-art results on eleven natural language processing tasks at the time of release. BERT was created to address the limitations of existing unidirectional language models. For sentence-level inference, this means they only look at text from left to right. In the self-attention layer, however, BERT enables tokens to attend to both sides. This is one of the major reason for its high performance. The most fascinating characteristic of BERT is how simple it is to utilize for a wide range of NLP tasks. The idea is to take a pre-trained BERT model and fine-tune it for the task at hand. The pre-trained model is unsupervised trained on a huge corpus, thus it learns the generic representations of the tokens from a vast corpus of text. Because the model comes pre-trained with a vast amount of context regarding language, grammar, and semantic representations, fine-tuning it for any other NLP task is simple

3.5 Iteration 4 : ML Model Deployment

When we develop a machine learning model, we need to think about how to deploy it. In this section, we will focus on building a pipeline for deploying our machine learning model for email classification. We used the dataset to build a prediction model that will accurately classify the received emails. After Training our model we saved in an h5 file the weights of our neural network architecture. So when we will use our model to predict we won't have to train it again each time. We developed a Flask api where we load the weight of our model and deployed it to a heroku app after we defined the requirements and our Procfile .

3.6 Iteration 5 : Extension Development

In this iteration we will address the way the chrome extension is developed in the project. It is a special web interface with special methods.
First, we start by creating the “Manifest.json” which is necessary for the chrome browser. The name of the plugin, descriptions, the project’s version, the icons we’ll use, permissions, and chrome APIs we’ll use are all specified in "manifest.json."
The User Interface is designated as "action" and the workers are designated as "service workers." We’ll go through each of these components and their functions in the next sessions.

Conclusion

In this chapter, we introduced the methodology included in this project, as well as the various evaluation metrics that will be used and the deep learning strategies that will be tested in the iterations. A timeline for each iteration is also given. Now that all is in place, we can focus on the configuration and outcomes of each iteration.

Chapter 4

Deep learning Approach and Results

Plan

Introduction

The results of the multiple phases are discussed in this chapter. The project was explored progressively, beginning with data discovery and analysis. Then progressing to the implementation of LSTM deep learning architecture,and then fine tuning of language models on dataset. The results are discovered after doing experiments and testing with multiple methods and parameters in order to find the optimal parameters for achieving high performance.

4.1 System Description

Our solution mainly focus on producing a pipeline to automate email categorization. First, our system collect emails, from email inbox to create our dataset and since we all know there might be several languages in one email inbox but here we mainly focused on English since it is the most used language for professional communications.

Second, Data cleaning is the next step of our pipeline, real world data always has noise, for emails it contains for example headers that contains bcc and cc users and it also have irrelevant non-textual data which we won't need in our process. In addition preprocessing strategy was needed to normalize our data.

Third, since our data does not have any labels for classification so instead of manually labeling our emails we suggested a system that extract keywords from emails content using a function that returns the most frequent words in emails and then for dimension reduction and topic generation we used LDA algorithm that helps as group similar keywords in one topic and for the same purpose we tried another approach where we used Doc2Vec to extract a signature to our Document and then trained a DBScan model so we can cluster emails per content and generate labels.

Finally, we trained two models an LSTM model where we were able to run multiple experiments and came up with final model architecture that we will detail later in the section below. We also run multiple experiments with the language model BERT which we fine tuned it to match with our requirements in this project.

The figure 4.1 below details our system including the deployment and extension development phases.

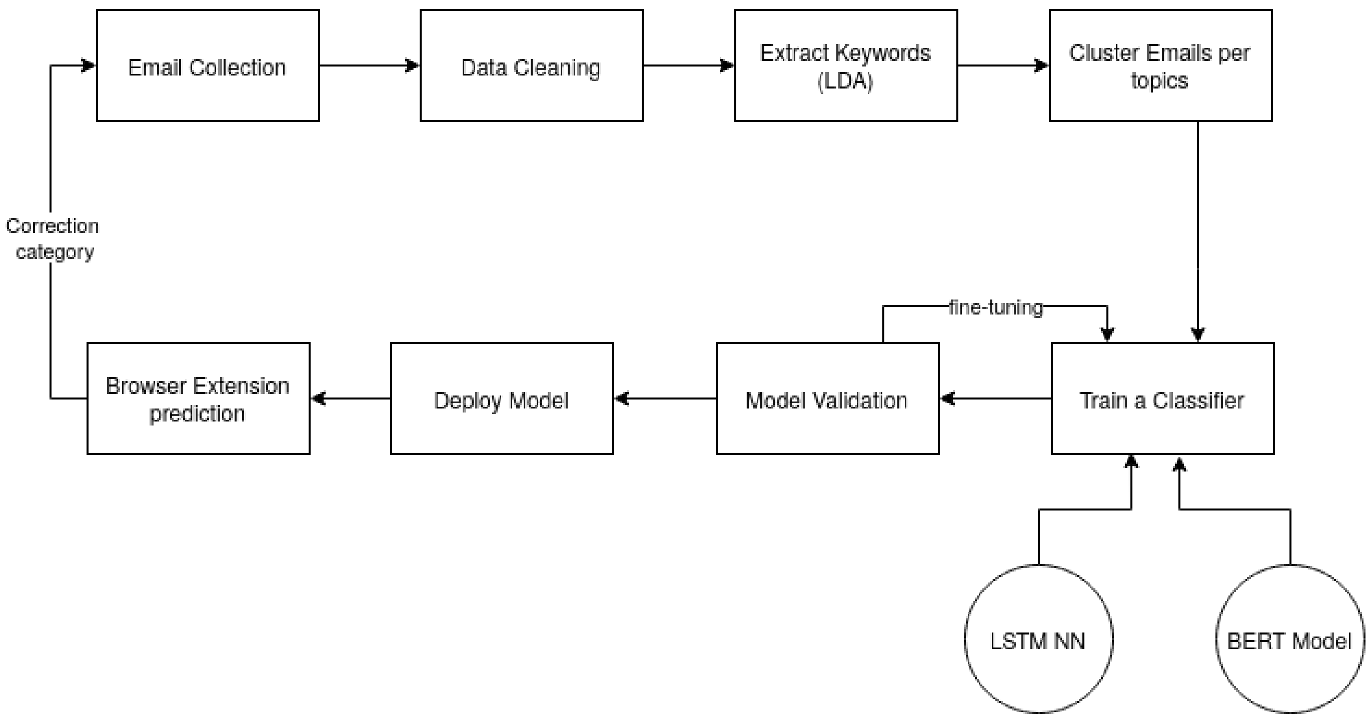


Figure 4.1: Pipeline Email Categorization

4.2 Data Description

During this project, the Enron Dataset was used. At first we thought of collecting our own dataset but we wanted to compare our results with previous work so mainly that's why we chose the Enron Dataset.

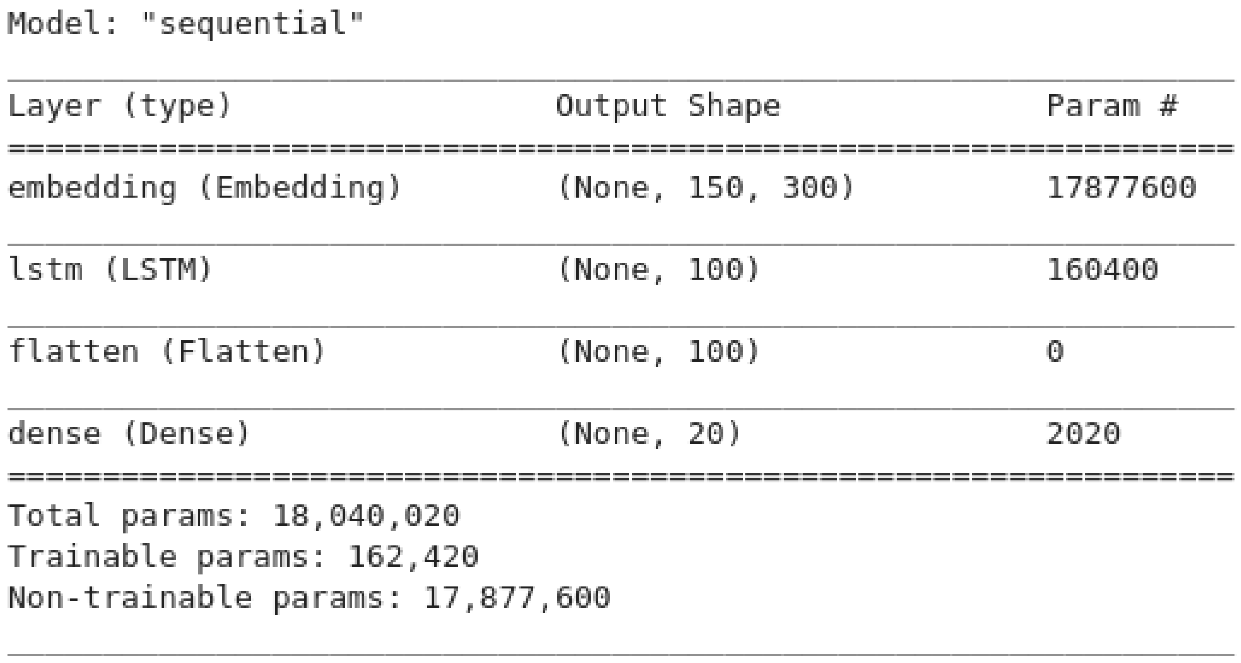
This dataset was collected and prepared by the CALO Project (A Cognitive Assistant that Learns and Organizes). It contains data from about 150 users, mostly senior management of Enron, organized into folders. The corpus contains a total of about 0.5M messages. This data was originally made public, and posted to the web, by the Federal Energy Regulatory Commission during its investigation. The email dataset was later purchased by Leslie Kaelbling at MIT, and turned out to have a number of integrity problems. A number of folks at SRI, notably Melinda Gervasio, worked hard to correct these problems, and it is thanks to them (not me) that the dataset is available. The dataset here does not include attachments, and some messages have been deleted "as part of a redaction effort due to requests from affected employees". Invalid email addresses were converted to something of the form user@enron.com whenever possible (i.e., recipient is specified in some parse-able format like "Doe, John" or "Mary K. Smith") and to no_address@enron.com when no recipient was specified. [8]

The dataset has 517401 email and over 150 users. Due to our limited resources in this project we have chosen to work only with 20 000 email to reduce training time. We have split the data for training set 70% and testing set of 30%.

4.3 Experiments with LSTM Neural Networks

We used the following pipeline to create the LSTM model. We began by preparing the data. We used two approaches for word embedding extraction: the first was Google's Word2Vec model, and the second was Glove Word2Vec, which is Stanford's pre-trained word embeddings trained on Wikipedia data with 6 billion tokens and a 400,000 word vocabulary. Word2Vec is a collection of linked models for generating word embeddings. These models are 2-layers deep neural networks that have been trained to recreate word linguistic contexts.

Word2vec takes a huge collection of text as input and provides a vector space with hundreds of dimensions, with each distinct word in the corpus allocated to a matching vector in the space. The Word2vec model produced embeddings with a maximum sequence length of 150 letters and a 300-dimensional vector. Our dataset was separated into training and testing with a 30 % testing split, and we split the training set into training and validation with a 10% split for doing validation during training. The artificial recurrent neural network (RNN) architecture known as long short-term memory (LSTM) is a type of recurrent neural network. Because LSTM contains feedback connections, it can analyze whole sequences of data, such as text sequences, as well as single data points. For the layers of our LSTM Neural network architecture we chose the following Architecture in the figure4.2



```
Model: "sequential"
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
embedding (Embedding)        (None, 150, 300)          17877600
_________________________________________________________________
lstm (LSTM)                  (None, 100)               160400
_________________________________________________________________
flatten (Flatten)            (None, 100)               0
_________________________________________________________________
dense (Dense)                (None, 20)                2020
=================================================================
Total params: 18,040,020
Trainable params: 162,420
Non-trainable params: 17,877,600
_________________________________________________________________
```

Figure 4.2: LSTM Neural Networks Layers Architecture

In our model, LSTM acts as a feature extractor of input sequence of email text and fully connected layer is used for classification. The LSTM model consisted of one LSTM layer with 100 units each and dropout of 0.3, 2 hidden layers with 64 units each and one softmax layer. The number of units in the softmax layer depends on the number of classes in the classification, Here we have 20 output class which means 20 Email categories and it is is used at the end to output probabilities of all classes. The model was trained using Adam optimizer with categorical cross entropy loss, learning rate of 0.01 and 60 epochs, and Accuracy as our metric of measure. 4.3,4.4

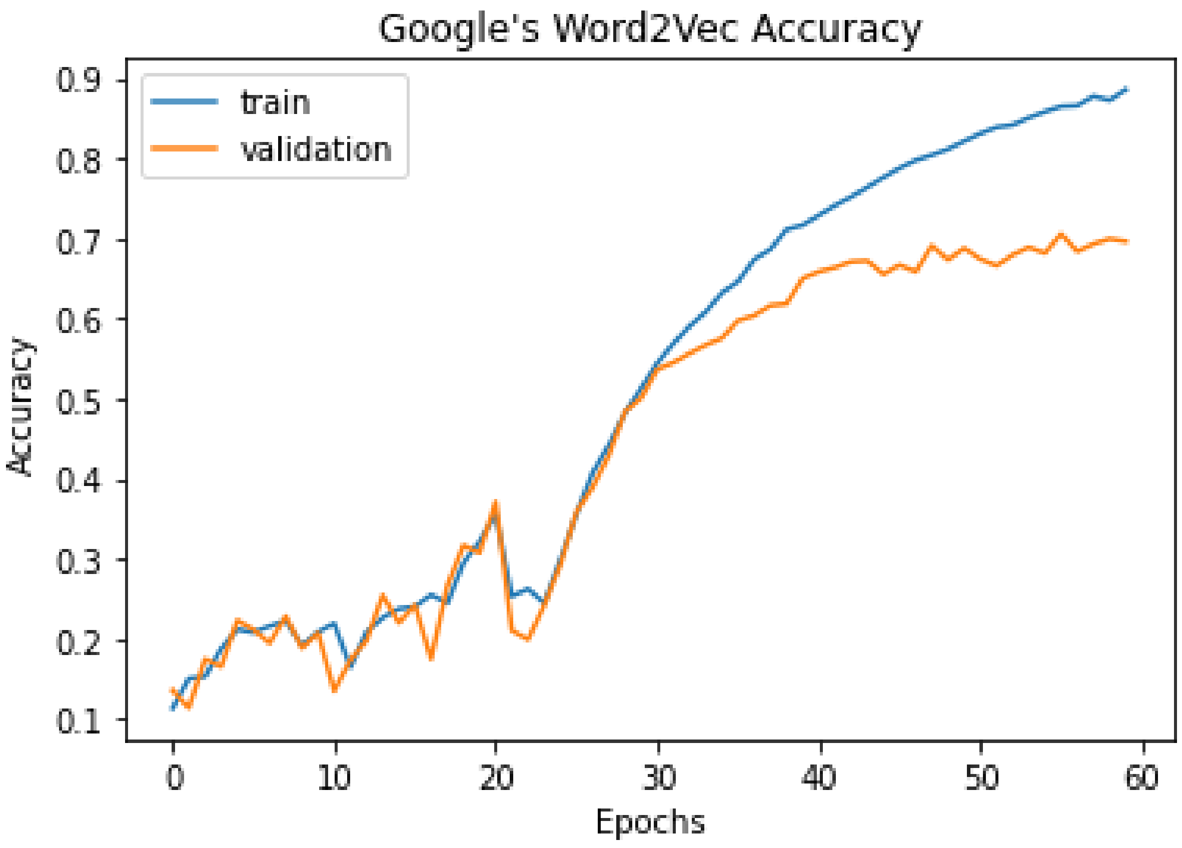


Figure 4.3: Google's Word2Vec embedding with LSTM model Accuracy

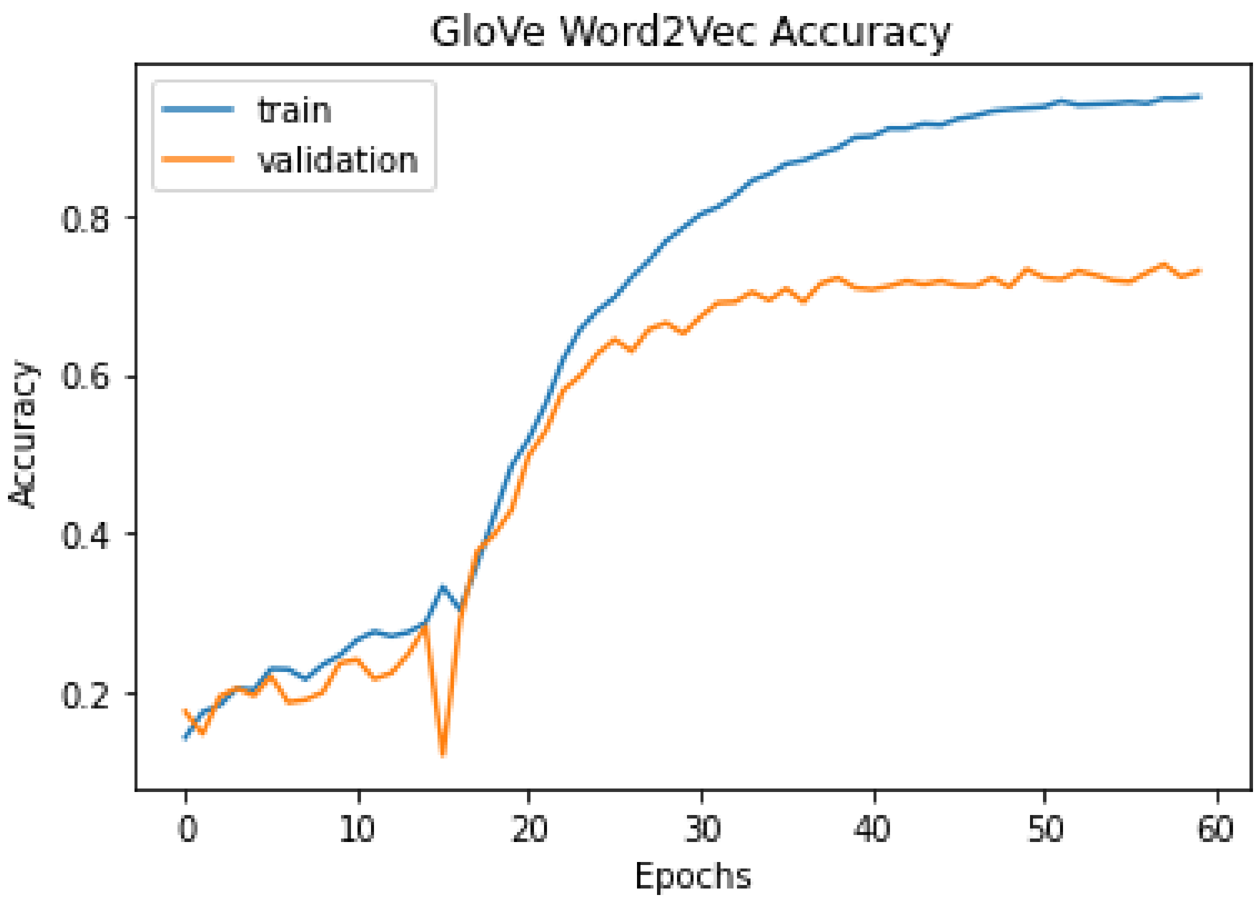


Figure 4.4: Glove Word2Vec embedding with LSTM model Accuracy

4.4 Experiments with BERT base uncased

Transfer learning was used to create the classification model. We will use NLP techniques in this method to learn the contextual meaning of emails. The current state-of-the-art deep learning model is the Bidirectional Encoder Representations from Transformers (BERT), which is a transformer encoder representation employed for modern NLP tasks. The BERT model is effective in determining the context of a text. The use of Transformer's bidirectional training to language modeling is BERT's major technological breakthrough.

The experiments using the pre-trained BERT base uncased are presented in the table 4.1 The best results achieved with Bert-base-uncased are 0,7776 using 3e-3 learning rate ,32 batch size ,128 sequence length and 10 epochs .

Table 4.1: Experiments with BERT Base Uncased

Model	learning rate	batch_size	sequence-length	epoch	Accuracy
Bert base uncased	1,50E-05	32	128	2	0.5513
Bert base uncased	5,00E-05	32	128	5	0.5978
Bert base uncased	3,00E-05	32	128	5	0.6046
Bert base uncased	3,00E-03	32	128	10	0.7776
Bert base uncased	5,00E-03	32	128	10	0.7210
Bert base uncased	4,00E-03	32	128	5	0.7506
Bert base uncased	4,00E-03	32	128	10	0.7704
Bert base uncased	2,00E-01	32	128	5	0.4710
Bert base uncased	5,50E-03	32	128	10	0.7105
Bert base uncased	2,50E-03	32	128	10	0.7397
Bert base uncased	2,00E-03	32	128	10	0.7278
Bert base uncased	3,50E-03	32	128	5	0.6982

4.5 Results and Discussion

As a result for the previously mentioned experiments the BERT model outperformed the LSTM models as shown in the following table 4.2

Table 4.2: Results

Embedding	Classifier	Accuracy
Google's Word2Vec	LSTM	0.689156
Glove's Word2Vec	LSTM	0.716879
BERT	BERT	0.777618

When compared to the other feature extractors utilized in the project, BERT extracted features produced the best results. We may conclude from the data that BERT derived features provided greater contextual meaning than any other feature extractors in our project. BERT model can extract text specific characteristics with which classification model can accomplish prediction job with greatest accuracy due to bidirectional sequence reading, multi-headed self attention layers, and layer normalization utilized in BERT model.

4.6 Model Deployment

Real-time predictions could be generated by deploying a deep learning model as a service. To do so we chose to use Flask to develop the API and Heroku. Salesforce offers Heroku which is a Platform-as-a-Service solution. It is supported by Amazon Web Services (AWS), and all Heroku apps and services are hosted on AWS. AWS provides the infrastructure and manages all load balancing, resource usage, networking, logging, and monitoring, while Heroku serves as a middleman, providing a flexible, autonomous fast deployment environment with all cloud features. Flask will provide a user interface for testing, and it can be coupled with the browser extension we'll create in the future chapter.

We followed 5 steps to deploy our application:

- We Saved the model in h5 file Create We created the Flask file "api.py" that loads the deep learning model and do predictions for the received data from our UI and it returns the class category of the text email in a JSON object.
- Create requirements.txt to setup Flask web app with all python dependencies.
- Create Procfile to initiate Flask app command
- Push all the previously mentioned files to repository in Github.
- Finally, we create an app on the Heroku dashboard after login and link it to the Github repository.

However The disadvantage recognized here that Heroku and Github offers a size limits for file and since our model is a deep learning model and the file containing the weights is over the size limits we weren't able to continue Deployment process.

Conclusion

A comparative analysis was conducted during this part. We studied Neural networks and preciously LSTMs and we conducted to the result that BERT outperformed the LSTM model and that's due o the mechanism of contextual embedding that BERT uses. Also we provided the API that will be consumed by the browser extension developed and explained in the next chapter.

Chapter 5

Email Extension Development

Plan

5.1 Introduction

This section will address the way the chrome extension development of the project. It is a special web interface with special methods . The plugin was coded in HTML for the structure , CSS/BOOTSTRAP for styling and JavaScript/JQuery for the logic and the DOM manipulation .

5.2 The User Interface (UI) :

The UI refers to the Interface that the users interacts with . As mentioned in chapter 3 , the UI is the "plugin.html" file for the structure adding to it some CSS/BOOTSTRAP for styling and JavaScript for the logic . The interface is pretty simple it consists of :

- Responsive Navigation Bar that has two items: "Home" and "How to use?" and each time we click on one of the navigation bar pills the content of the page will change responsively .
- The "Home" page content consists of a main container that has a text area where we are going to put the text of the email we want to classify . Adding to that there is two buttons "Classify Email" and "Clear".
- The "Clear" button clears the value of the text area .
- The "Classify Email" button send a message to background worker with the text area value and wait for the response .
- The "How to use?" page is just a page with a demo video so that the users know how to use the plugin .

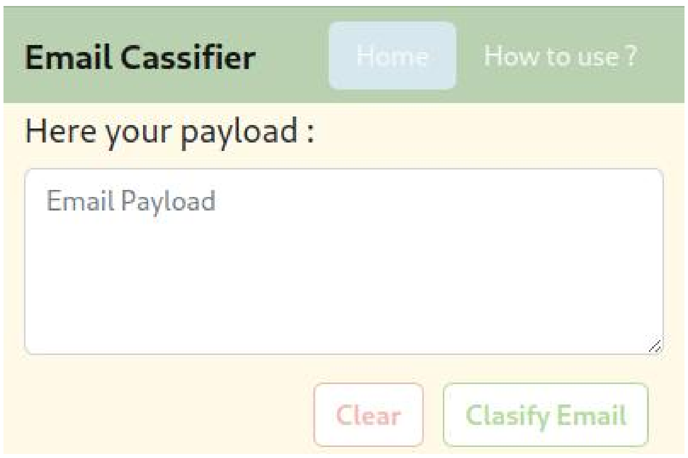


Figure 5.1: Representation of The UI

5.3 The Service Worker :

The workers are specified in “background.service_workers” tag as "eventPage.js" .
It consists of some JavaScript logic that is written as controllers to handle browser events (such as navigation to page , sending API requests,handling context menus) not user events (clicking buttons/navbar items. . .).
We created a context menu in the that allows us ,after selecting the text of the email, to send it directly to the text area instead of just copying the text content and pasting it .
Also, after clicking on "Classify Email" in the "plugin.html",the worker get a real-time message to send a “POST” request to the server and get the response of the prediction for us .
Once the worker get a response it send it back to the UI pack (exactly to "plugin.js" so this JavaScript code will handle the response from the "eventPage.js")

5.4 Communication between UI , Worker and the server :

After selecting the text of an email and sending it through the context Menu , the background and the UI are two separate environments that need to communicate .
We use the chrome storage API that allows us once the text of an email gets selected we store its value in storage .
Each time we open the extension popup the UI searches in the storage if there is an existing text or it got cleared .
But , not like context , if the “Classify Email” button got triggered in the UI we need to fetch data and return the response in real time that is why we used another chrome API called messages because messages API is a realtime API .
Each time the “Classify Email” is clicked we send a message from the UI to the Worker with the text area value then the worker gets the message and sends the post request to our server and waits for the response .
Once the server returns the response it is sent back to the UI through the same message API .
But after getting the response the user must know the type of email he wants to classify that’s why why used another chrome API named "Notifications"
We specify the title of the notification , its icon and the message of the notification as the class of the email .

After specifying the notification options we send back to the user a notification telling him the class of the text of the email he sent .

Getting everything together this diagram represents our project components and how they communicate together :

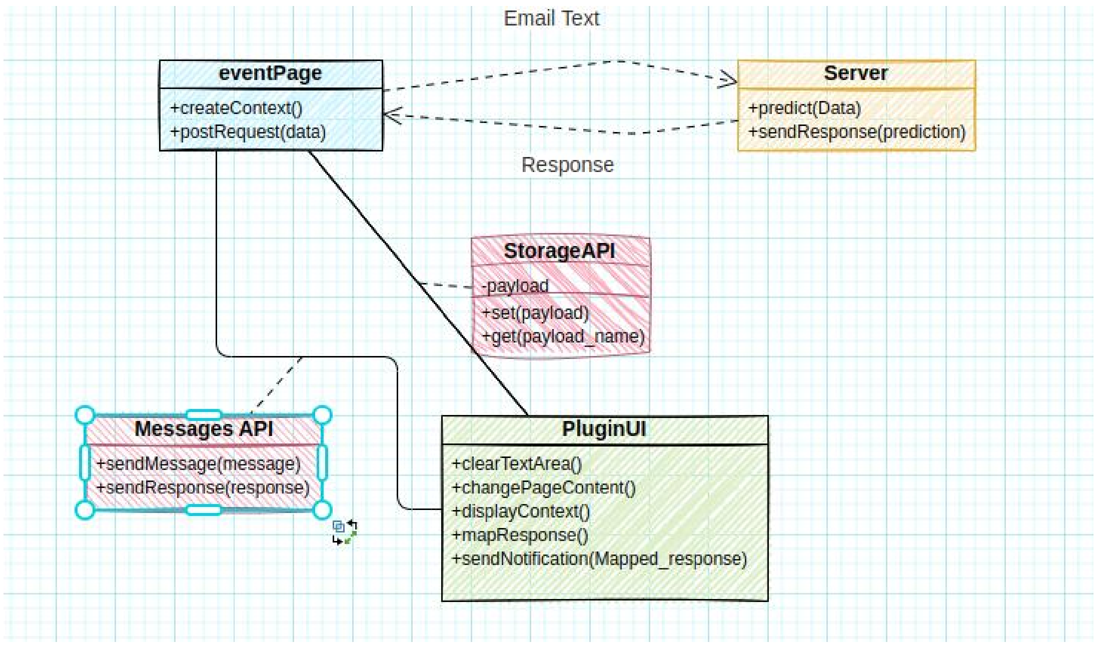


Figure 5.2: Interaction between Components

Conclusion

In this chapter,we talked about the process of developing the chrome extension and its phases from creation the required file "manifest.json" to creating the "popup.html" and styling it with CSS and BOOTSTRAP and finally we added some JavaScript logic to both the User Interface so we manipulate DOM elements responsively and the service worker that works in the background in isolated environment to create a context menu and request middleware between the web server and the UI .

We also explained clearly how each of these two files communicate with each other through some chrome APIs and communicate with the web server we made to predict the email class .

General Conclusion

For a multitude of reasons, email is a common means of communication in today's culture. It is a terrific method to market things, and it is the most cost-effective way for businesses to communicate. Email can be delivered instantaneously, and it is ideal for keeping records since it leaves a virtual trail of who sent it, what it was about, and what it included.

These days, it's common to have many email addresses: personal emails, business emails, and school / college emails. Most people receive between 20 and 40 emails every day. It might be a lot more for some. It takes a long time to go through each and every email to determine if it's important. A need for an automated system to Categorize email is urgent because with the mas of emails received or sent everyday.

NLP is used for a variety of tasks, including text classification, which may be used to classify emails. The models are initially trained using a dataset known as the training dataset. Following that, these models can recognize each email which category belongs to automatically.

This project tackles this sort of issue by offering a Browser Extension that shows a notification within the class of which category the user email belongs to.

Multiple relevant research and articles concerning NLP in general, and especially on email Classification , were examined to obtain these results, as were numerous preprocessing approaches and potential experiments with many pretrained language models and deep learning models.

This project provided an excellent chance to evaluate the recent Google's pre-trained model BERT, the model that has broken several limits on the amount of how well it can handle language-based tasks.

This project enabled me to become proficient in the state-of-the-art library for NLP tasks Transformers and its several pretrained models.

I also had the opportunity to experiment with simple Transformers and text processing techniques, as well as develop my skills to a real-world situation.

Also, during this project we learned about ML Deployment pipeline and how it can be automated, how a model in production can contribute to improve the result obtained. Developing the browser extension was challenging to us and we learned a lot during the process.

For future work we will focus on improving our results as mentioned earlier we will integrate a field in our UI where user can correct the class to help us label our dataset and we will re-trian our models with the new inputs. Also we might conduct other studies such as Graph Neural networks where I believe that it may produce better results.

Bibliography

[1] M. Rhanoui, M. Mikram, S. Yousfi, and S. Barzali, “A cnn-bilstm model for document-level sentiment analysis”, *Machine Learning and Knowledge Extraction*, vol. 1, pp. 832–847, Jul. 2019. DOI: 10.3390/make1030048.

[2] K. Weiss, T. Khoshgoftaar, and D. Wang, “A survey of transfer learning”, *Journal of Big Data*, vol. 3, May 2016. DOI: 10.1186/s40537-016-0043-6.

[3] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, *Attention is all you need*, Jun,2017. arXiv: 1706.03762 [cs.CL].

[4] T. Wolf, L. Debut, V. Sanh, *et al.*, “Transformers: state-of-the-art natural language processing”, in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, Online: Association for Computational Linguistics, Oct. 2020, pp. 38–45. [Online]. Available: https://www.aclweb.org/anthology/2020.emnlp-demos.6.

[5] J Alammar, *The Illustrated Transformer [Blog post]*, https://jalammar.github.io/illustrated-transformer/, 2018.

[6] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: pre-training of deep bidirectional transformers for language understanding*, Oct,2018. arXiv: 1810.04805 [cs.CL].

[7] Y. Liu, M. Ott, N. Goyal, *et al.*, *Roberta: a robustly optimized bert pretraining approach*, 2019. arXiv: 1907.11692 [cs.CL].

[8] “Enron email dataset”. (), [Online]. Available: https://www.cs.cmu.edu/~./enron/.

2, Abou Raihane Bayrouni 2080 l'Ariana **Tél** : 71 706 164 **Fax** : 71 706 698 **Email** : isi@isim.rnu

ملخّص

في الوقت الحاضر ، فإن أكثر وسائل الاتصال الاحترافية استخداماًا هي البريد الإلكتروني الذي يتم تلقي كمية هائلة من البريد الإلكتروني منه كل يوم. هذا يتطلب تطوير نظام تصنيف البريد الإلكتروني الآلي. نظراً لقيود الموارد ، لم يعالج أي امتداد لتصنيف البريد الإلكتروني الاكتشاف الآلي للفئات اعتماداً على المحتوى النصي للبريد الإلكتروني. يتم تنفيذ عمليات معالجة اللغة الطبيعية)مراجعة وتصنيف رسائل البريد الإلكتروني هذه. هذا المشروع عبارة عن مهمة اكتشاف فئة باستخدام أحدث تقنيات التعلم العميق ودراسة أحدث طراز من ل زضة.

كلمات مفاتيح : مجموعة بيانات البريد الإلكتروني زضصد ، تصنيف البريد الإلكتروني، NLP ، BERT ، التعلم العميق ، امتداد المتصفح.

Résumé

De nos jours, le moyen de communication professionnel le plus utilisés est l'email d'où la quantité énorme d'email reçut chaque jour. Cela nécessite le développement d'un système automatisé de categorisation des emails. En raison des ressources limitées aucune extension pour la classification des emails n'a abordé à ce jour la détection automatisée des catégories dépendant du contenu textuel de l'email. Des opérations de traitement du langage naturel (NLP) sont effectuées afin d'examiner et de catégoriser ces emails. Ce projet est une tâche de détection de catégorie en utilisant les techniques les plus récentes en deep learning et en faisant l'étude du modele du Google le plus récente BERT.

Mots clés : ENRON Email Dataset,Catégorisation des emails, NLP, BERT, Deep Learning, extension de navigateur.

Abstract

Nowadays, the most used means of professional communication is the email from which the huge amount of email received each day. This requires the development of an automated email categorization system. Due to limited resources no extension for the classification of emails has so far addressed the automated detection of categories depending on the textual content of the email. Natural Language Processing (NLP) operations are performed to review and categorize these emails. This project is a category detection task using the latest deep learning techniques and studying the Google model of the latest BERT.

Keywords : ENRON Email Dataset, Email Categorization, NLP, BERT, Deep Learning, Browser Extension.