الجمهورية الجزائرية الديمقراطية الشعبية

وزارة التعليم العالي والبحث العلمي

جامعة سعيدة د. مولاي الطاهر

كلية العلوم

قسم: الإعلام الآلي

Masters thesis

Speciality: Computer Security and Cryptography

Theme

Spam detection using BERT

Presented by :

Abderrahmane Khial

Ahmed Oussama Hakkoum

Dirigé par :

Dr. Aissa Fellah

Promotion 2023 - 2024

ملخص

ملخص باللغة العربية

في عصرنا الحالي و مع التقدم الذي وصلنا اليه . اصبح البريد الرقمي ركيزة اساسية في التواصل و المعاملات بين المؤسسات الضخمة و الأشخاص , هذا ما أدى إلى ارتفاع نسبة الهجمات الالكترونية و الاختراقات عن طريق الرسائل غير مرغوب فيها و التي تعتبر تهديدا لخصوصية المستخدم و واحد من أكبر مشاكل الأمن السيبراني. فمن هنا يجب استخدام طرق أكثر تطورا لمحاربة المخترقين والمحتالين .إحدى أحدث هذه الطرق هي استخدام الذكاء الإصطناعي و تحديدا BERT والذي حقق قفزة نوعية في مجال فهم معالجة اللغة الطبيعية و التي يعتمد عليها في فهم و تحليل النصوص مما يعطي قدرة وأداء أفضل في الكشف عن الرسائل المزعجة والضارة . و قد تنافس الكثيرون في تحسين و زيادة كفاءة أنظمة الكشف عن الرسائل غير مرغوب فيها , نتطرق هذه أطروحة الى كيفية استخدام BERT و DistilBERT للحصول على دقة أعلى ونتائج أفضل. بالإضافة إلى إدماج الذكاء الصناعي في تطبيق ويب للتحقق من رسائل مشكوكة و API للمبرمجين للإستفادة من خدماته في تطبيقاتهم .

الكلمات المفتاحية : البريد الرقمي , الهجمات الإلكترونية, الرسائل غير مرغوب فيها, خصوصية المستخدم, مشاكل الأمن السيبراني , الذكاء الاصطناعي ,معالجة اللغة الطبيعية ,أنظمة الكشف عن الرسائل غير مرغوب فيها ,BERT , DistlBERT, تطبيق ويب,API,

Abstract

Your summary in English

In our present time and with the progress we reached. Email has become an essential pillar of communication and transactions between large enterprises and people alike , which has led to a higher rate of cyberattacks and hacks through unwanted messages that are a threat to user privacy and one of the biggest cyber security challenges. Hence, more sophisticated methods must be used to fight hackers and scammers . One of the latest methods is the use of artificial intelligence, specifically BERT, which has made a significant leap in understanding natural language processing and on which it relies in understanding and analyzing text, thus giving better ability and performance in detecting unwanted and harmful messages. Many have competed to improve and increase the efficiency of spam detection systems , this thesis discusses usage of BERT and DistilBERT to get higher accuracy and better results. In addition, our model is integrated into the web application to check suspicious messages and an API for developers to use its services in their applications.

Keywords : email, cyberattacks ,unwanted messages ,cyber security challenges ,user privacy ,artificial intelligence ,BERT ,DistilBERT ,natural language processing ,spam detection systems , web application ,API

Résumé

Votre résumé en français

Dans notre temps actuel et avec les progrès que nous avons réalisés. L' e-mail est devenu un pilier essentiel de la communication et des transactions entre les grandes entreprises et les individus à la fois, ce qui a résulté en un taux plus élevé de cyber-attaques et de piratages par messages indésirables qui constituent une menace pour la confidentialité des utilisateurs et l'un des plus grands défis de la cybersécurité.

Par conséquent, des méthodes plus sophistiquées doivent être utilisées pour lutter contre les pirates et les escrocs. L'une des méthodes les plus récentes est l'utilisation de l'intelligence artificielle, en particulier BERT, qui a fait un saut significatif dans la compréhension du traitement du langage naturel et sur laquelle elle s'appuie pour comprendre et analyser les textes, donnant ainsi une meilleure capacité et des performances dans la détection de messages indésirables et nuisibles.Beaucoup ont concouru pour améliorer et augmenter l'efficacité des systèmes de détection de spam. Cette thèse traite de l'utilisation de BERT et DistilBERT pour obtenir une plus grande précision et de meilleurs résultats. En outre, notre modèle est intégré dans l'application web pour vérifier les messages suspects et une API pour les développeurs d'utiliser ses services dans leurs applications.

Mots clés : email, cyberattaques, messages indésirables, défis de cybersécurité, confidentialité des utilisateurs, intelligence artificielle, BERT, DistilBERT, traitement du langage naturel, systèmes de détection de spam, application Web, API

شكر وتقدير

الحمد لله حمدا كثيرا حتى يبلغ الحمد منتهاه والصلاة والسلام على أشرف مخلوق أناره الله بنوره واصطفاه.

وانطلاقا من باب من لم يشكر الناس لم يشكر الله نتقدم بخالص الشكر والتقدير للأستاذ **فلاح عيسى** على إرشاداته وتوجيهاته التي لم يبخل بها علينا يوما , كما نتقدم بجزيل الشكر الى كل يد رافقتنا في هذا العمل من قريب او بعيد .

كما لا ننسى أن نشكر جميع الأساتذة الذين تتلمذنا على ايديهم و أخذنا منهم الكثير.

إهداء

أحمد الله تعالى على منه و عونه لي لإتمام هذا البحث.

أهدي هذا البحث الى :

إلى من أُفضِّلها على نفسيو من وضع المولى - سبحانه وتعالى -الجنة تحت قدميها، ووقَّرها في كتابه العزيز أمي الغالية و إلى من وجوده سبب سعادتي وفلاحي في الدنيا والأخرة أبي الحبيب **خيال عبد الحميد**, وإلى اخوتي الذين كانو معي في سراء و ضراء **خيال محمد البشير و خيال خ**, كما ولا ننسى أفراد عائلتي الأحباء.

و إلى أصدقائي الذين كانو لي سندا **عفان إلياس ,حكوم أحمد أسامة, حمادي جلالي , بلقاسمي صدام حسين و ميلس رياض وباقي اصدقاء.**

والى اخوتنا في في فلسطين الجريحة"اللهم فك اسرهم وسدد رميهم وتقبلهم مع الصالحين"

عبد الرحمان

الحمد الله حمدا كثيرا طيّبا مباركا يليق بعظمته و جزيل فضله على وضعي في هذا الطريق وتوفيقي لهذا البحث,

نشكره على ما أنعم علينا في مسعانا, اللهم لك الفضل في الأولى والآخرة.

اما بعد , أهدي هذا العمل إلى :

كل من كابد الصعوبات و قهر المستحيلات من أجلي , إلى الذي أرفع به رأسي و به أفتخر أبي العزيز حكوم بن عامر , و إلى من أوصى بها نبينا و سيدنا محمد صلى الله عليه و سلم ثلاثا أمي الغالية .

إلى أخواتي اللاتي كن سندا لي في حياتي .

إلى أصدقائي و أولهم رفيقي خيال عبد الرحمان , عفان إلياس ,عبد الحاكم عبد الجليل , ميلس رياض , مسعودي عبد الرحمان , باقي الاصدقاء .

إلى أشقائنا الجرحى في فلسطين. ونسأل الله أن يفك الحصار عنهم، وأن ينصرهم على عدونا وعدوهم. وأن يرحم موتاهم و يفرغ عليهم صبرا و يرزقهم فرجا قريبا ان شاء الله .

أحمد أسامة

Table of abreviations

A:

AI : Artificial Intelligence
API : Application Programming Interface
Adam : Adaptive Moment Estimation

B:

BERT : Bidirectional Encoder Representations from Transformers
BN: Batch Normalization

C:

CNN : Convolutional Neural Network
CVS : Comma Separated Value

D:

DP: Dropout
DL : Deep Learning

G:

GLUE : General Language Understanding Evaluation

K:

KNN: K-Nearest Neighbor

L:

LLM : Large Language Model

M:

ML : Machine Learning
MLM : Masked laguage Modeling

N:

NLP : Natual Language Processing
NER : Named Entity Recognition

R:

ReLU : Rectified Linear Unit

S:

SVM: Support Vector Machine

Table of content

Chapter One

Introduction

Chapter 1: Introduction

1.1 PREAMBLE

In an era where digital communication is crucial, cyber security has emerged as a serious concern. Email, being an irreplaceable communication channel, is one of the most exploited vectors by cyber-attackers. Among the numerous challenges that troubles email communication, spam stands out as a persistent and significant threat. Spam not only suffocate users with unwanted messages but often serves as a conduit for malicious activities, posing severe risks to both individuals and organizations.

Spam is unsolicited and irrelevant messages, and it is a pervasive issue. According to a 2023 report by Kaspersky [1], spam constituted nearly 45.6% of global email traffic, highlighting the scale of the problem. The nature of spam emails varies widely, from harmless advertisements to specially engineered phishing schemes designed to deceive recipients into divulging sensitive information. The ubiquity and variety of spam underscore the necessity for robust and effective spam detection mechanisms.

Over the years, various approaches have been developed to combat spam, ranging from primitive keyword-based filters to sophisticated machine learning algorithms for instance Gupta and Bakliwal et al[21] discussed machine learning techniques for spam detection on SMS ,Thaer Sahmoud and Dr. Mohammad Mikki [26] implemented an effective high-performance bert-based spam detector. A recent and notable advancement in this domain is the introduction of Bidirectional Encoder Representations from Transformers (BERT)[14]. Developed by Google, BERT represents a significant leap in the field of natural language processing (NLP). As it excels at understanding the context of words within a sentence, making it exceptionally effective for a wide array of NLP [7] tasks, including spam detection.

While we are not the first to apply BERT to spam detection, our research aims to make significant contributions to this evolving field. By leveraging BERT's powerful contextual understanding, we seek to enhance the accuracy of spam detection systems. Our objective is to provide a more refined and effective solution to the spam problem, therefore improving the security and reliability of email communication.

1.2 PROBLEMATIC

Despite the advancements in spam detection technologies, spam remains an unsolved hustle in email communication field. Existing solutions have yet to reach perfection, as spammers continually evolve their techniques to bypass filters. This relentless evolution of spamming strategies necessitates ongoing innovation and improvement in spam detection methodologies.

1.3 OBJECTIVES

1. Develop a BERT-based spam detection system: Utilize BERT's contextual understanding to identify and filter spam emails more accurately.
2. Experiment with various configurations: test different configurations and training strategies to optimize BERT for spam detection.
3. Integrate the model in a web application: offer the capabilities of the model to the end-user to test suspicious email content.
4. Create a practical API: implement the developed system into an accessible API for real-world application and evaluation.
5. Contribute new insights: offer innovative experimental enhancements to the field of spam detection.

1.4 STRUCTURE OF THE MANUSCRIPT

Chapter 1 Background: This chapter will provide a comprehensive review of existing spam detection techniques, covering both traditional methods and modern machine learning approaches. Setting the stage for the proposed BERT-based approach.

Chapter 2 Approach: This chapter will delve into the details of the proposed spam detection technique using BERT. It will outline the data preprocessing steps undertaken to prepare the data for model training. Then we discuss the model selection and the proposed model architecture and other hyperparameter value selection.

Chapter 3 Experimentation: This chapter will focus on the experimental setup and evaluation of the proposed BERT-based spam detection system. It will describe the evaluation metrics used, such as precision, recall, and F1-score.and compare the obtained results to related work in the same context (datasets) .Finally, will cover the integration of the developed spam detection model into a web application and the creation of a practical API. It will describe the design and implementation of the web application, including user interfaces and functionalities.

Chapter 5 Conclusion and Future Work: The final chapter will summarize the research contributions, findings, and limitations of the study. It will highlight the significance of the proposed BERT-based approach and its potential impact on improving spam detection accuracy. Additionally, this chapter will provide suggestions and recommendations for future research directions, potentially exploring areas for further refinement or expansion of the proposed technique.

Chapter Two
Background

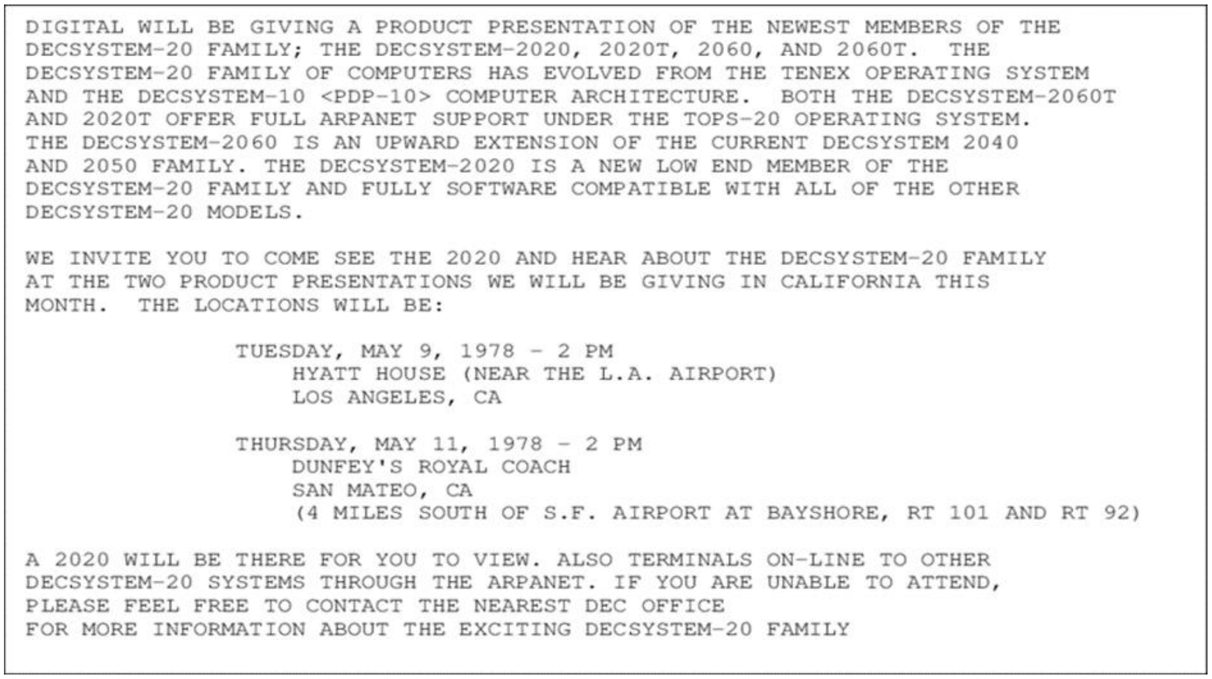
Chapter 2 : Background

2.1 Introduction :

In this chapter, we will discuss the concepts necessary for understanding the rest of our work, in particular the notion of spam and AI tools for spam detection and finally related work (in particular with Bert).

2.2 Spam Detection :

The term came from a sketch by the British comedy group Monty Python in which the word "spam" was repeated constantly, similar to the repetitive nature of unsolicited messages.
As for the context of emails, it generally represents unsolicited emails containing commercial advertisements, “get rich fast” scams, or phishing content. Some feelings the spammers exploit are: greed, trust, helpfulness, and impose or urgency [2]. The first documented spam electronic mail (not yet known as email) was sent to several hundred ARPANET users on May 3, 1978. It was an advertisement for a presentation by Digital Equipment Corporation for their DECSYSTEM-20 products, sent by Gary Thuerk, one of their marketers.



```
DIGITAL WILL BE GIVING A PRODUCT PRESENTATION OF THE NEWEST MEMBERS OF THE
DECSYSTEM-20 FAMILY; THE DECSYSTEM-2020, 2020T, 2060, AND 2060T.  THE
DECSYSTEM-20 FAMILY OF COMPUTERS HAS EVOLVED FROM THE TENEX OPERATING SYSTEM
AND THE DECSYSTEM-10 <PDP-10> COMPUTER ARCHITECTURE.  BOTH THE DECSYSTEM-2060T
AND 2020T OFFER FULL ARPANET SUPPORT UNDER THE TOPS-20 OPERATING SYSTEM.
THE DECSYSTEM-2060 IS AN UPWARD EXTENSION OF THE CURRENT DECSYSTEM 2040
AND 2050 FAMILY. THE DECSYSTEM-2020 IS A NEW LOW END MEMBER OF THE
DECSYSTEM-20 FAMILY AND FULLY SOFTWARE COMPATIBLE WITH ALL OF THE OTHER
DECSYSTEM-20 MODELS.

WE INVITE YOU TO COME SEE THE 2020 AND HEAR ABOUT THE DECSYSTEM-20 FAMILY
AT THE TWO PRODUCT PRESENTATIONS WE WILL BE GIVING IN CALIFORNIA THIS
MONTH.  THE LOCATIONS WILL BE:

               TUESDAY, MAY 9, 1978 - 2 PM
                   HYATT HOUSE (NEAR THE L.A. AIRPORT)
                   LOS ANGELES, CA

               THURSDAY, MAY 11, 1978 - 2 PM
                   DUNFEY'S ROYAL COACH
                   SAN MATEO, CA
                   (4 MILES SOUTH OF S.F. AIRPORT AT BAYSHORE, RT 101 AND RT 92)

A 2020 WILL BE THERE FOR YOU TO VIEW. ALSO TERMINALS ON-LINE TO OTHER
DECSYSTEM-20 SYSTEMS THROUGH THE ARPANET. IF YOU ARE UNABLE TO ATTEND,
PLEASE FEEL FREE TO CONTACT THE NEAREST DEC OFFICE
FOR MORE INFORMATION ABOUT THE EXCITING DECSYSTEM-20 FAMILY
```

Figure 1.1 : First Spam email ever

2.2.1 Why is it a problem?

Spam poses a substantial challenge in every aspect of digital communication due to its capacity to spread malicious content, such as malware and deceptive hyperlinks. Email traffic jams result in reduced efficiency and financial burdens. Cybercriminals leverage spam as a means to execute fraudulent schemes, thereby aggravating financial damage and weakening confidence in digital communications. This highlights the criticality of implementing efficient spam mitigation and detection strategies.

Statistics :
According to kaspersky [1] lab In December 2023, 45.6% of all emails were spam, scattered across different categories [3]:

- ➢ Marketing/Advertising : 36%
- ➢ Adult content : 31.7%
- ➢ Financial matters : 26.5%
- ➢ Others (fraud,scams,miscellaneous): 5.8%.

- ✧ 94% of all malware is delivered by email
- ✧ The average cost of BEC exploits was $5.96 million in 2021.
- ✧ spam costs businesses $20.5 billion annually in decreased productivity.
- ✧ For every 12,500,000 emails sent, spammers receive one reply [4].

2.3 AI & Bert

2.3.1 AI

Advancements in big data, cloud computing, artificial neural networks, and machine learning have enabled engineers to construct machines capable of replicating human intelligence. Computers can utilize these technologies to achieve artificial intelligence (AI), enabling them to detect, recognize, learn, respond to, and solve issues. Undoubtedly, this intelligent technology will inevitably revolutionize the future work environments. AI is currently considered by many as a driver that is important to the fourth industrial revolution, and it may launch the fourth revolution in education. The integration of AI into the school curriculum has already commenced. So artificial intelligence (AI) is a dynamic discipline of computer science focused on constructing machines that can execute operations previously needed by human intellect.

2.3.2. Machine learning

Machine learning is a branch of artificial intelligence that relies on algorithms and models that learn from a set of data until they are able to give predictions without the need for clear programming, unlike previous systems that rely on previously defined rules. Patterns and connections in machine learning algorithms are identified through examples or experiments, which gives them the ability to adapt to new and unusual circumstances. Machine learning has different types, including supervised, semi-supervised, unsupervised, and reinforcement learning, in order to adapt to tasks and different types of data. Features are essential for classification tasks, and algorithms are trained to automatically extract and select features, which advances the classification process, leads to technological breakthroughs and enables data-driven decision-making and innovation.

2.3.3 Deep learning

Deep learning is a branch of machine learning that seeks to imitate human thinking and behavior independently of any human intervention or support. It employs multi-layer neural networks, which are done by automatically extracting features via architectural sculpting using hyperparameters. With the rapid development and greater availability of large datasets, deep learning techniques provide more relevant results compared to traditional machine learning methods. Deep learning provides a great ability to model and understand complex patterns in data, which gives more accurate predictions and automates activities such as natural language processing, autonomous systems, and recognition. sound or images, and this is what made it a cornerstone of contemporary artificial intelligence.

2.3.4 - LLM (Large language models)

A large language model (LLM) is a type of artificial intelligence (AI) program that can recognize and generate text, among other tasks. LLMs are trained on huge sets of data, hence the name "large." LLMs are built on machine learning: specifically, a type of neural network called a transformer model. [5].

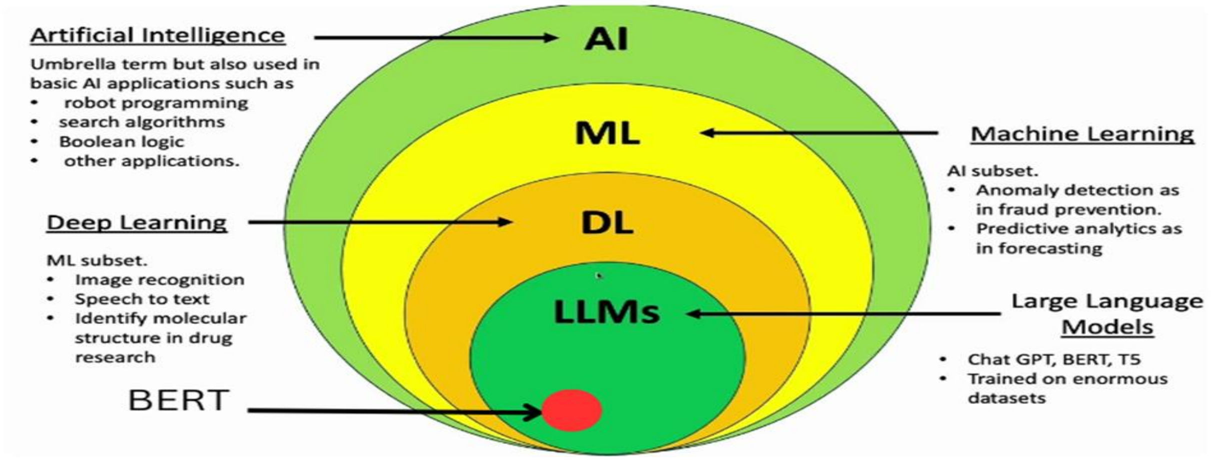


Figure 1.2: components of AI ,with adaptation by [6]

2.3.5 - BERT

2.3.5.1 - Natural Language Processing (NLP)

NLP or Natural Language Processing, is a discipline that focuses on the understanding, manipulation, and generation of natural language by machines. Thus, NLP is really at the interface between computer science and linguistics. It is about the ability of the machine to interact directly with humans [7].

2.3.5.2 - Neural network

A neural network is a computer model that draws inspiration from the structure and functioning of the human brain. It is composed of linked layers of nodes, also known as neurons. Every individual neuron in the network receives input data and performs computations on it, transmitting the outcome to the subsequent layer. This process enables the network to acquire knowledge about patterns and generate anticipations. A neural network consists of an input layer of neurons (or nodes, units), one or two (or even three) hidden layers of neurons, and a final layer of output neurons. Figure (1.3) shows a typical architecture, where lines connecting neurons are also shown. Each connection is associated with a numeric value called a weight, which determines the strength of the signal between neurons. Additionally, each neuron has a bias, a numeric value that is added to the input of the neuron before applying the activation function. Weights and biases are adjustable parameters that the network learns during the training process to minimize error and improve accuracy. [8]

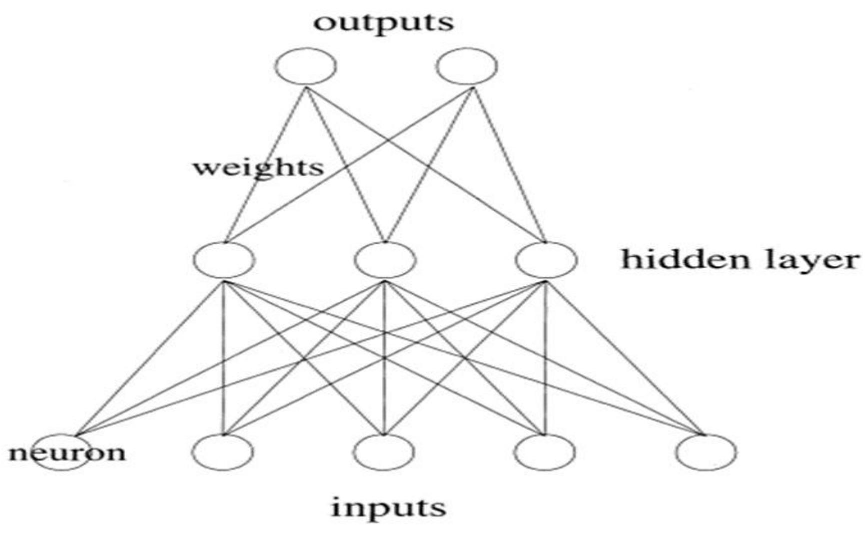


Figure 1.3 : a simple neural network

2.3.5.3 - Transformers

Transformers are a type of deep learning model [9]. They have revolutionized Natural Language Processing (NLP) by enabling models to process text more efficiently and effectively than previous architectures. Transformers have a simpler architecture without having any convolutional or recurrent layers. Transformer models outperformed the current models with less training cost. Many transformers based models have been invented :K-BERT [10], XLNet[11].

A. Self-attention

Self-attention is the mechanism used to determine the interdependence of tokens in the given input sequence. Self-attention encodes a token by taking information from other tokens. It consists of three weight matrices query, key, and value vectors learned during the training process. Multiheaded self-attention is the extension of self-attention, consisting of multiple sets of the query, key, and value vectors built into transformers . However, this entire process will be managed by the transformers library .

B. Parameters and hyper-parameters

Parameters are the intrinsic variables of a machine learning model that are learnt from the training data. These settings are modified throughout the training phase to reduce the error and increase the model's performance.
Hyperparameters are the external settings established before the training process starts. Unlike parameters, hyperparameters are not learnt from the data but are given by the user to influence the learning process.

C. Activation functions

Activation functions help determine the neural network's output with the help of some non-linear function to the corresponding output of neurons.

ReLU:

The rectified linear activation function, often referred to as ReLU, is a linear function that outputs the input directly if it is positive, and zero otherwise.

$$f(x) = max\ (0, x)$$

Softmax:

The softmax is a function fits perfectly well in the final layer of a neural network model that is designed for a classification problem. It serves to transform raw output scores into probabilities. This process means that the obtained values range within $0 < P < 1$ and their sum totals 1, which makes them easily comprehendable probabilities.

$$\frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}}$$

D. Loss Function

Cross-entropy loss quantifies of how wrong the model is. It increases when the predicted probability departs from the true label. So predicting 0.012 when the true label is 1 would be undesirable and result in a high loss value. An ideal model should resulting to have a log loss of 0.

$$\textbf{cross entropy loss} = -(y\log(p) + (1-y)\log(1-p))$$

E. adam optimizer

The Adam optimizer is a widely utilized technique for training deep learning models. Adam employs adaptive learning rates for individual parameters and integrates momentum by keeping an exponentially declining average of previous gradients. As a consequence, this leads to effective and resilient training, rendering it extensively utilized in diverse machine learning and deep learning applications.

F. attention head

An attention head is a component of the multi-head attention mechanism within the Transformer architecture. Each attention head independently computes a weighted sum of input values (using a set of learned weights), which allows the model to focus on different parts of the input sequence. By using multiple attention heads, BERT can capture various aspects of the relationships between words, leading to a more comprehensive understanding of the context and meaning in natural language processing tasks.[9]

2.2.5.4 BERT Architecture:

One of the most popular transformer-based models, is an encoder stack of transformer structure and applies the bidirectional training of transformers to language modeling [12]. BERT designs include significant feedforward networks and attention heads. It takes a classification token (CLS) and a word sequence as input. Each layer uses self-attention and passes the result through a feedforward network to the next encoder.

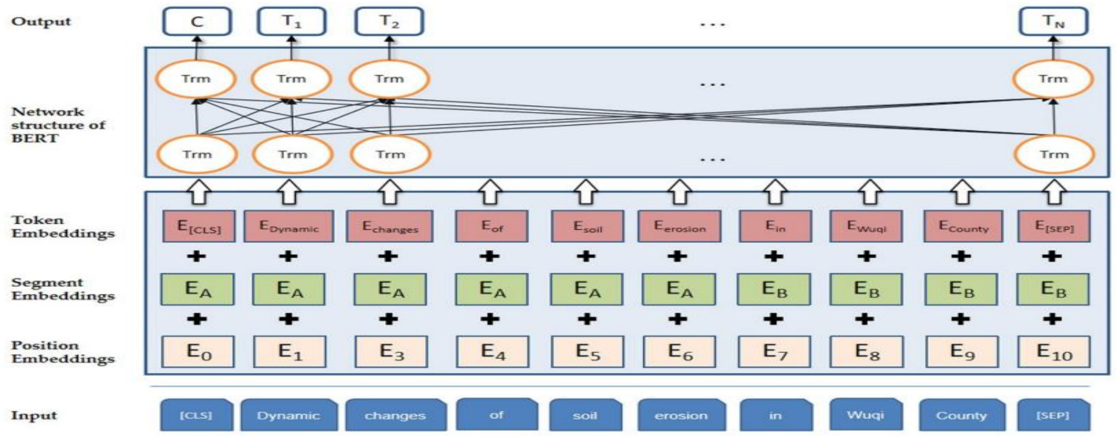

Output
C
T1
T2
...
TN
Network structure of BERT
Trm
Token Embeddings
E[CLS] EDynamic Echanges Eof Esoil Eerosion Ein EWuqi ECounty E[SEP]
Segment Embeddings
EA EA EA EA EA EA EB EB EB EB
Position Embeddings
E0 E1 E3 E4 E5 E6 E7 E8 E9 E10
Input
[CLS] Dynamic changes of soil erosion in Wuqi County [SEP]

Figure 1.4 : The overall structure of the BERT model adapted from [13]

The BERT architecture is based on the multilayer bidirectional transformer described in [12]. The authors trained two versions of the neural network a standard one with 12 layers and 768 coordinates in the view (110 million trained parameters in total) and a large one with 24 layers and 1024 coordinates (340 million parameters). BERT uses text embeddings described in 2016 to represent an input sequence

BERT was tested on several standard datasets to compare its performance with other published models. So, on the GLUE test (General Language Understanding Evaluation) , a set of tasks and datasets that test natural language comprehension), the BERT-based model showed an average superiority of 4.5% and 7% (for standard and large neural networks, respectively) compared to the best-known models. [12]

A.Encoder

The basic BERT model contains an encoder with 12 transformer blocks, 12 attention areas, and a textual representation dimension of 768. The model receives a text sequence of no more than 512 tokens as input and outputs its vector representation. The sequence can contain one or two parts, devided by a special token , each of which necessarily begins with the token and has a special classification representation [14].

B.Pre-training model

Pre-trained models revolutionized the field of natural language processing (NLP) by being a building block for many language related tasks, BERT in particular has emerged as a leader in this class due to his ability to capture bidirectional context from text data. The process of creating a pre-trained model involves training a deep learning network on huge datasets for any task, usually language understanding. For instance, BERT has been trained on the whole English Wikipedia corpus and Book Corpus to finally yield a model capable of understanding language context[12].

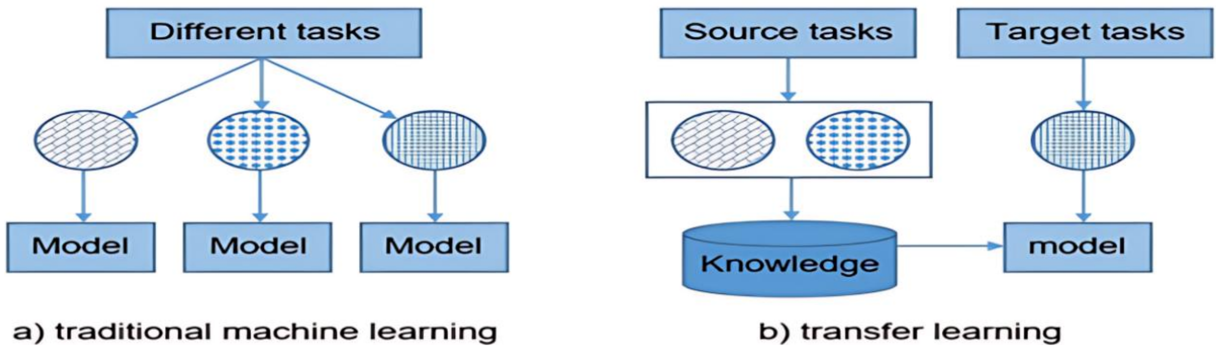


Figure 1.5: Difference between learning transfer and traditional ML, adapted from [15]

Using a pre-trained model means knowledge gained from a task, like language understanding, is used for other tasks (named-entity recognition (NER), sentiment analysis, text classification, etc). In the case of spam detection, fine-tuning BERT on labeled datasets of spam and ham messages allows the model to adapt its pre-learned knowledge to the detection task. This approach greatly reduces training time and computational costs while improving results.

C. Training a pretrained

Masked language modeling

also known as the MLM , is widely used technique to train text oriented models and consists of teaching a model to predict a word (token) in a text sequence, replaced by a special token [MASK].During training, 15% of the tokens in each input sequence are randomly selected . This technique is one of two BERT training tasks and is detailed in [16].

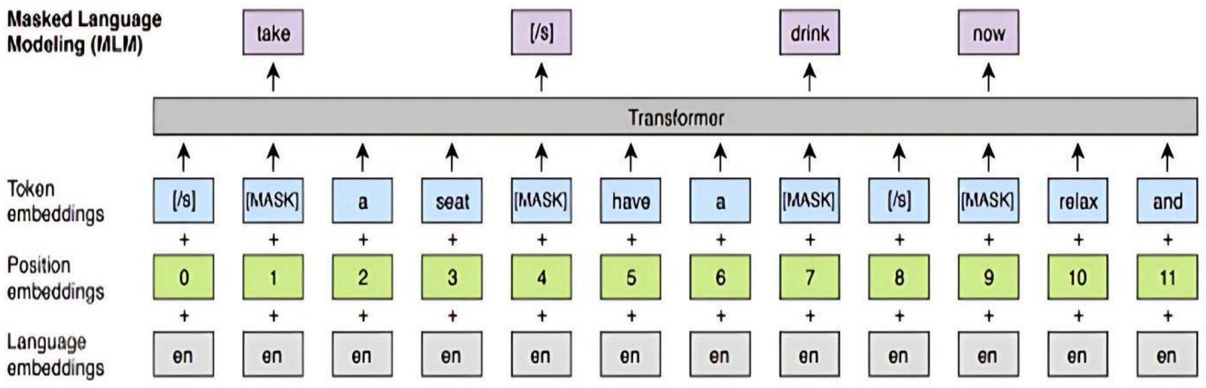


Figure 1.6 : Masked language modeling [16]

the next sentence prediction

Next Sentence Prediction (NSP) [12] is to determine if the two sentences are contiguous or not, or randomly selected. This task helps BERT in understanding the connection that exists between the two sentences. In pre-training, BERT uses two sentences at a time, where half of them are adjacent and the other half are two random sentences. The input is represented by embeddings and special tokens are added.

Fine-tuning

Fine-tuning involves adjusting a pre-trained model's parameters to fit a specific task, without using task-specific parameters initially. This method is used in models like OpenAI GPT, where the model is pre-trained and then retrained by fine-tuning all internal parameters. For BERT, the typical fine-tuning approach replaces the original output layer with a task-specific layer or adjusts the entire model. This includes learning new output layer parameters and modifying all original weights, such as word embeddings, Transformer blocks, and the pooler.[17]

2.3.5.5 DistilBERT

DistilBERT [18], is a distilled version of bert base . It uses knowledge distillation which is a compression technique that trains small model to reproduce the behavior of a larger model. The model has an architecture similar to the BERT base with 40% fewer parameters. DistilBERT runs 60% faster than the BERT model by preserving over 97% of BERT's performances. A limited set of attention patterns may often be repeated across different heads and cause the model to be over-parameterized. Distillation removes such parameters and improves the model performance.

2.4 Related work

2.4.1 Rule based

uses predefined criteria to identify and filter out spam emails. Those criteria are based on the common patterns, keywords and behaviors whose expressions contain, for example, "free money" or "click here", hyper usage of punctuation or/and suspicious email headers.

2.4.2 Machine learning

Zamil et al.[19](2019) proposed a method for distinguishing between ham and spam images using a combination of SVM and kNN, achieving an average accuracy of 97.27% when K equals 20. next , Nithesh Reddy[20] (2021) introduced a new spam detection method effective in distinguishing spam from its content, achieving an accuracy of 92% on average.

2.4.3 Deep learning

In 2018, Gupta and Bakliwal et al[21] took two different dataset of SMS and applied different machine leaning classifiers and both dataset's results support CNN for giving high accuracy. In their paper they discussed machine learning techniques for spam detection on SMS. After evaluation of different algorithm, they found that Convolutional Neural Network Classifier achieves the highest accuracy of 99.19% and 98.25% . Rodrigues and Fernandes et al(2022) [22] contributed their work for spam detection. The primary focus of their work was to detect spam from tweets. After extracting features from tweets, different classifiers were applied such as decision tree, random forest, Naive Bayes and deep learning methods. They have also performed sentiment analysis in tweets by using 1D convolutional neural network (CNN) model. After analyzing all mention models, they came to know that deep learning model (LSTM) has achieved hight accuracy (98.74%) out of all models.

2.4.4 bert :

Jie[23] discussed the bilingual language multi-type spam detection model using M-BERT, which used image-based spam detection and achieved an accuracy of about 96%. Lee's research [24] using the proposed CATBERT model by collecting phishing emails. The BERT model was also used for other applications like fake news detection, lie detector, sentiment analysis. Barsever and Singh [25] proposed a model with the new generative adversarial network to detect lies. he proposed a sentiment analysis algorithm based on BERT and Convolutional Neural Network with an accuracy rate of 93.6% .Thaer Sahmoud, Dr. Mohammad Mikki [26] have implemented an effective high performance spam detector that can detect spam emails or spam SMSs, they have trained their model on Enron corpus, SpamAssassin corpus, Ling-Spam corpus and SMS spam collection corpus, their spam detector performance was 98.62%, 97.83%, 99.13% and 99.28% respectively. C Oswald, SE Simon, A Bhattacharya [27] model ,DistilBERT+SVM (Poly) obtained an accuracy of 98.07%(SMS Spam dataset). VS Tida, S Hsu [28] provided different good results on various datasets. It can be helpful in the real-time scenario for spam classification using BERT based on the combination of different datasets as an input to the designed model. It achieved a 97% accuracy at a F1 score of 97%.

authors	year	Datasets	accuracy	F1-score
Jie [23]	2020	- Enron, - SMS Spam - TREC Spam Corpus - Lingspam	96%	Not mentioned
Barsever and Singh [25]	2020	private dataset	90.5%	Not mentioned
Thaer Sahmoud, Dr. Mohammad Mikki [26]	2022	-Enron -SpamAssassin -Ling-Spam -SMS Spam	98.62% 97.83% 99.13% 99.28%	Not mentioned
C Oswald, SE Simon, A Bhattacharya [27]	2022	-Enron Spam Dataset -SMS Spam -SpamAssassin -Ling-Spam Dataset	97.23% 98.07% 97.89% 98.45%	97% 97% 97% 98%
VS Tida, S Hsu [28]	2022	-SpamAssassin -Enron -Spam Text -Ling-Spam Dataset	98% 97% 98% 98%	97% 97% 93% 94%

Table 1.1 : Related work

2.5 conclusion:

In this chapter, we have explored the definition of spam, the role of Artificial Intelligence (AI) in enhancing spam detection, and the advanced capabilities of the BERT model in natural language processing. We reviewed the evolution of spam detection methods from rule-based approaches to machine learning, deep learning, and the state-of-the-art BERT-based techniques.

In the next chapter, we will delve into our approach and methodology for spam detection using BERT. We will also discuss the experimentation process, including the design, implementation, and evaluation of our model, to demonstrate its effectiveness and accuracy in identifying and filtering spam.

Chapter Three
Approach

Chapter 3: Approach

3.1 Introduction:

We have trained several BERT models using different BERT variants and architectures. In pursuit of a high-performance model that would still be agile and scalable for our real-time spam detection system, this chapter shall summarize the essential procedures for attaining that objective. The following flowchart demonstrates the process.

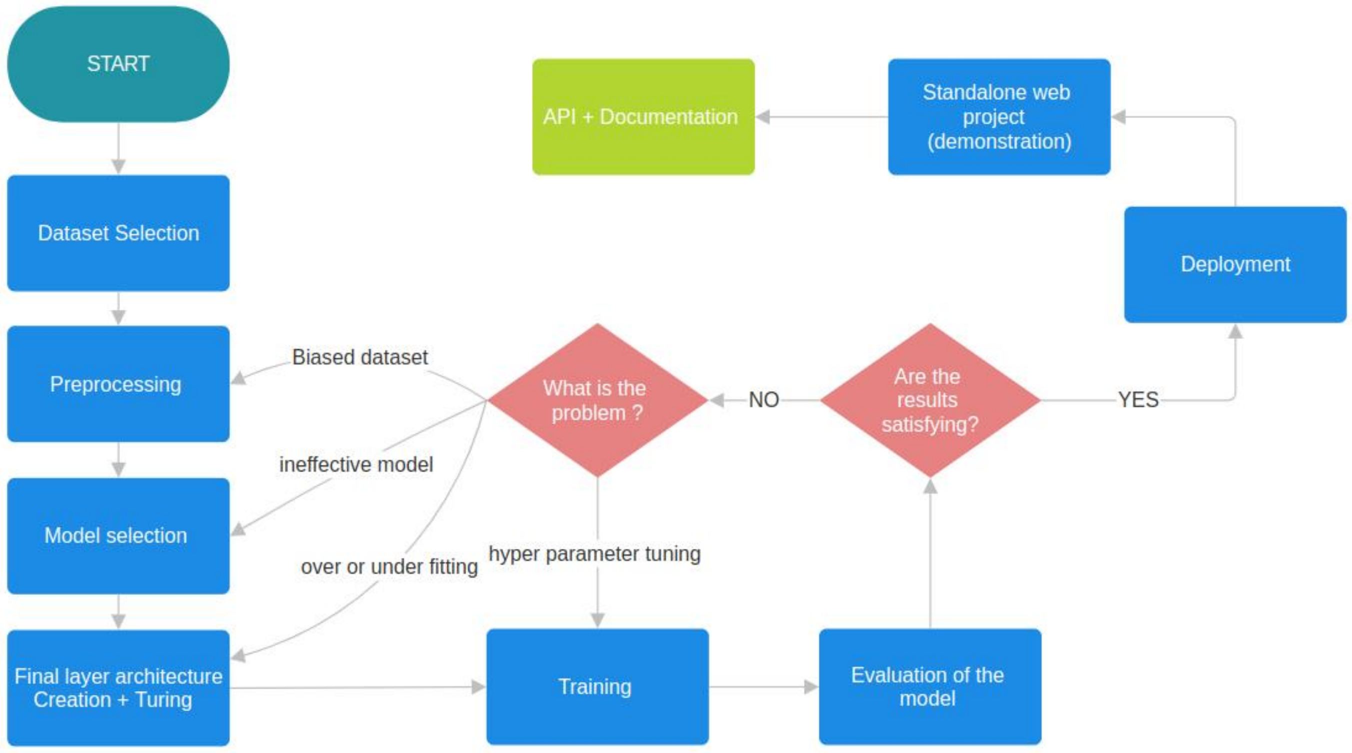


Figure 2.1: Workflow Visualization

3.2 Preprocessing:

Like every machine learning model, preprocessing is a vital aspect since it may considerably impact the performance of the model. In the first part of this chapter, we will review the measures that were taken in depth and underline their relevance.

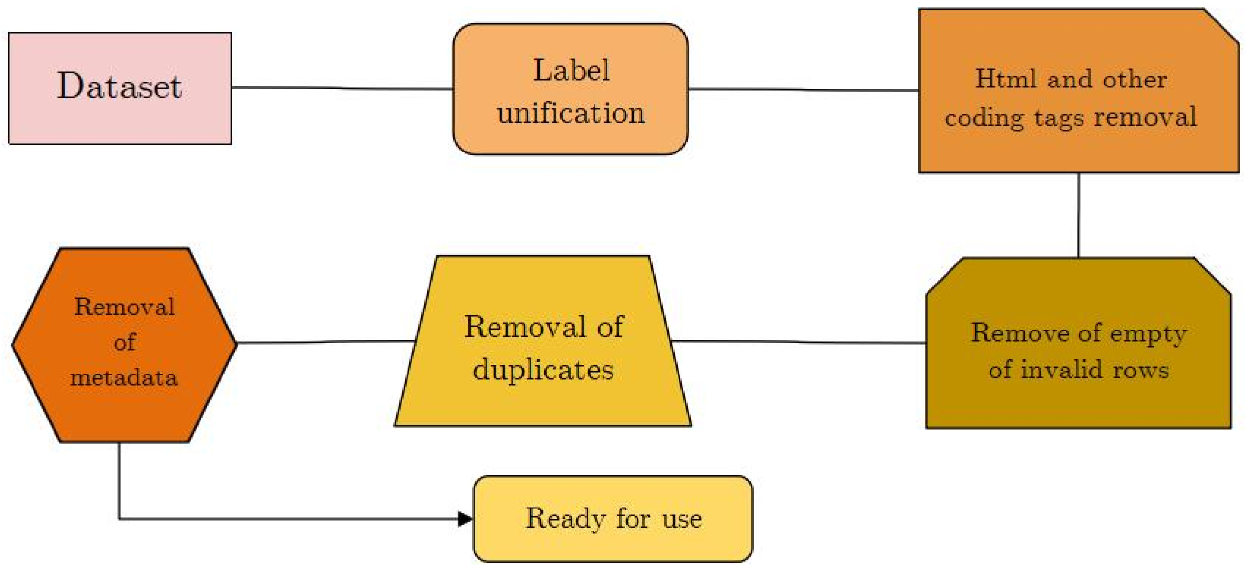


Figure 2.2: Preprocessing workflow

3.2.1 <u>Label unification and numerical representation:</u>

The first step involved mapping different label categories to a unified set of categories, ensuring consistency in labeling across datasets. Then we opted for a numerical representation of the labels (0 for ham and 1 for spam). We decided to use numerical labels because they offer compatibility with machine learning libraries, standardization across all datasets, and an increase in the interpretability of the model, and they facilitate debugging. Because of this, they are an option that is both practical and reasonable for data labeling. Figure 2.3 highlights an example of label unification and adapting numerical presentation.

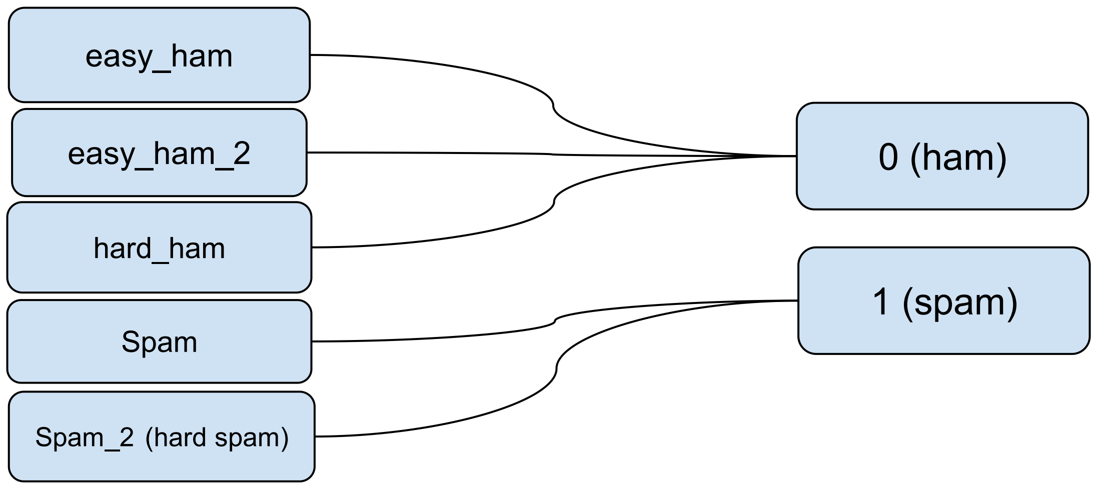
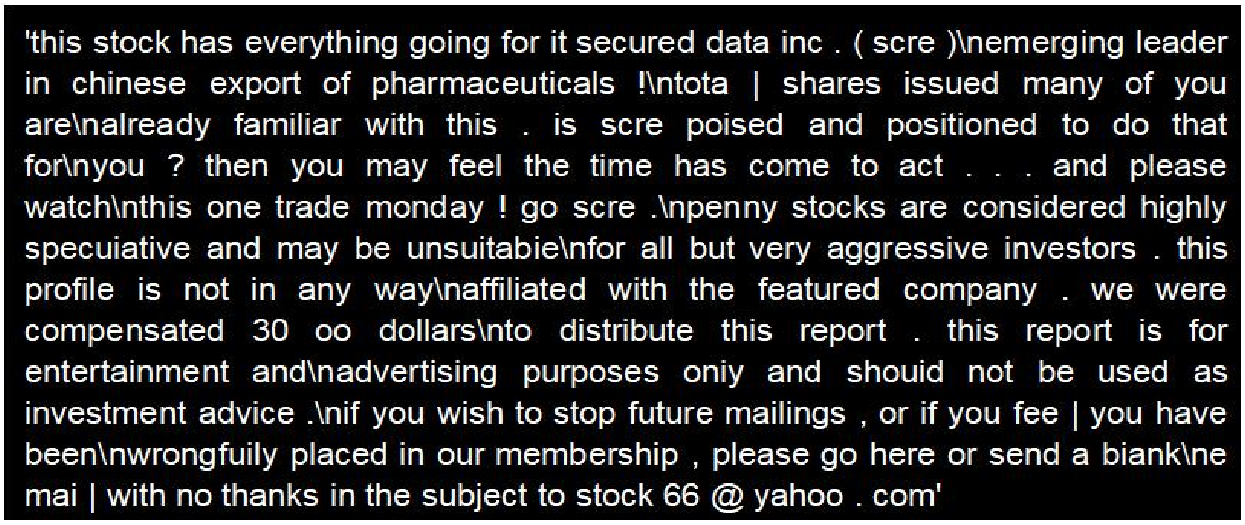


Figure 2.3: Example of Label Unification and Numerical Representation

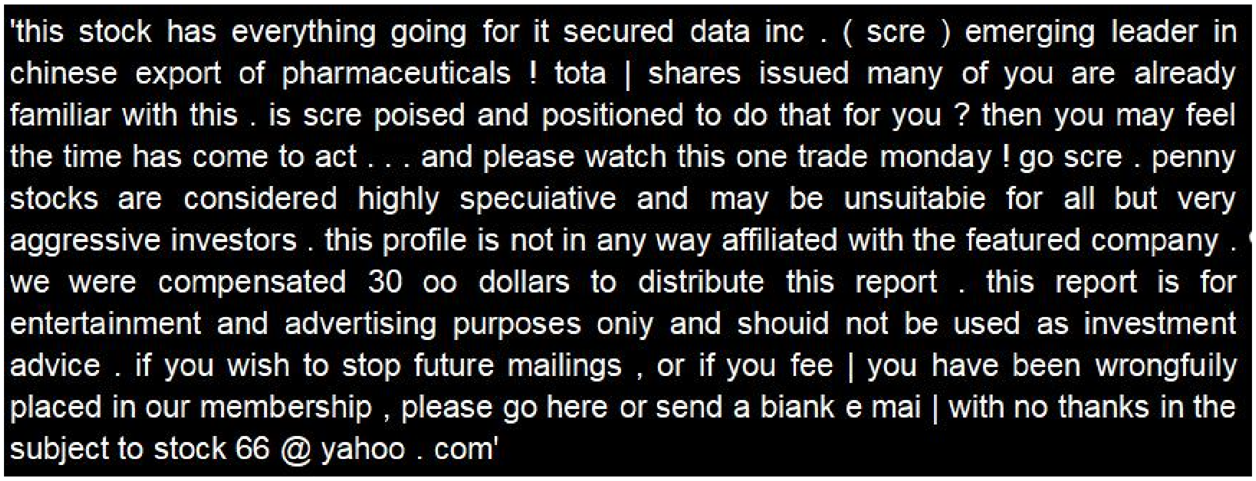
3.2.2 Html and Coding Tag Removal:

After that, we clean the content of the dataset from any html tags in it, such as (<br>, <a>, etc.) and other coding tags, like (\n ,\t ...). The goal of this step is to make the model receive the data like an end user would; there are no html or coding tags in our emails. This ensures integrity and helps the model recognize patterns in human readable data.



'this stock has everything going for it secured data inc . (scre)\nemerging leader in chinese export of pharmaceuticals !\ntota | shares issued many of you are\nalready familiar with this . is scre poised and positioned to do that for\nyou ? then you may feel the time has come to act . . . and please watch\nthis one trade monday ! go scre .\npenny stocks are considered highly specuiative and may be unsuitabie\nfor all but very aggressive investors . this profile is not in any way\naffiliated with the featured company . we were compensated 30 oo dollars\nto distribute this report . this report is for entertainment and\nadvertising purposes oniy and shouid not be used as investment advice .\nif you wish to stop future mailings , or if you fee | you have been\nwrongfuily placed in our membership , please go here or send a biank\ne mai | with no thanks in the subject to stock 66 @ yahoo . com'

Figure 2.4: Unprocessed email with leftover coding tags



'this stock has everything going for it secured data inc . (scre) emerging leader in chinese export of pharmaceuticals ! tota | shares issued many of you are already familiar with this . is scre poised and positioned to do that for you ? then you may feel the time has come to act . . . and please watch this one trade monday ! go scre . penny stocks are considered highly specuiative and may be unsuitabie for all but very aggressive investors . this profile is not in any way affiliated with the featured company . we were compensated 30 oo dollars to distribute this report . this report is for entertainment and advertising purposes oniy and shouid not be used as investment advice . if you wish to stop future mailings , or if you fee | you have been wrongfuily placed in our membership , please go here or send a biank e mai | with no thanks in the subject to stock 66 @ yahoo . com'

Figure 2.5: Preprocessed email

3.2.3 Invalid rows:

After that, we checked every row of the datasets for invalid or empty values and discarded them. This ensures data integrity and avoids the model being biased.

3.2.4 Metadata removal

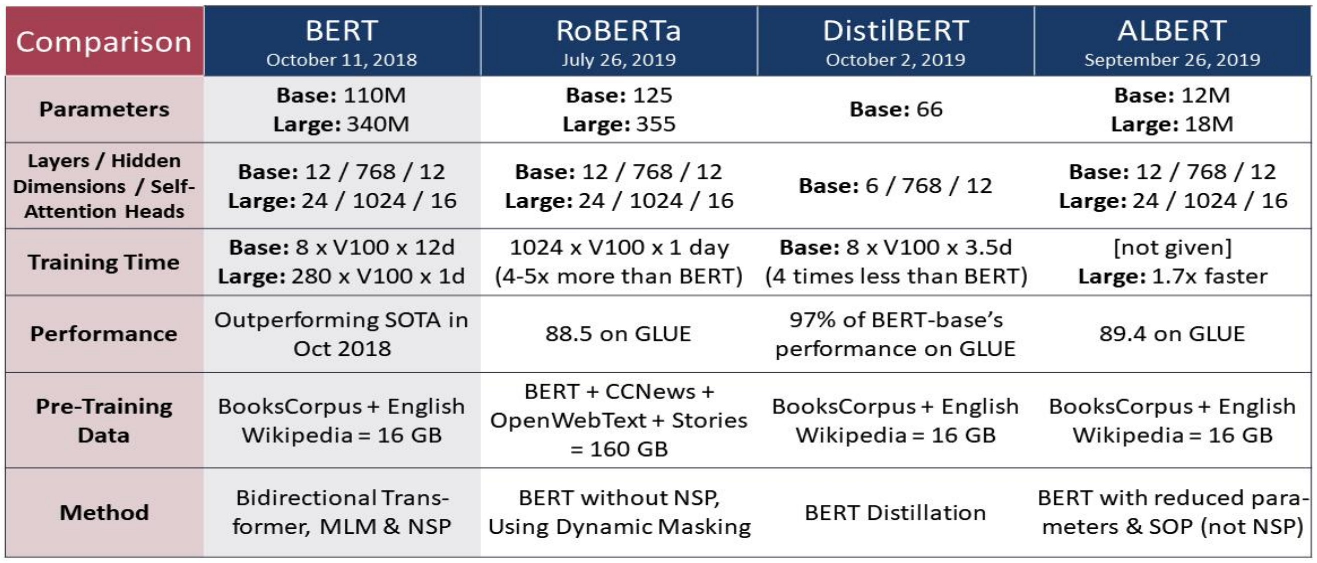
Finally, we removed the metadata header from email text, and we now have a clean, ready-to-use dataset.

These steps prepare the data for use in our model and remove any irrelevant bits, unwanted characters, and tags. In the upcoming chapter, we will look into the details of model selection and architecture creation.

3.3 Model selection:

3.3.1 BERT

Choosing a suitable BERT variant for our project was a deep search in itself because of the massive variety of variants, each with strong and weak traits.



Comparison	BERT October 11, 2018	RoBERTa July 26, 2019	DistilBERT October 2, 2019	ALBERT September 26, 2019
Parameters	**Base:** 110M **Large:** 340M	**Base:** 125 **Large:** 355	**Base:** 66	**Base:** 12M **Large:** 18M
Layers / Hidden Dimensions / Self-Attention Heads	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16	**Base:** 6 / 768 / 12	**Base:** 12 / 768 / 12 **Large:** 24 / 1024 / 16
Training Time	**Base:** 8 x V100 x 12d **Large:** 280 x V100 x 1d	1024 x V100 x 1 day (4-5x more than BERT)	**Base:** 8 x V100 x 3.5d (4 times less than BERT)	[not given] **Large:** 1.7x faster
Performance	Outperforming SOTA in Oct 2018	88.5 on GLUE	97% of BERT-base's performance on GLUE	89.4 on GLUE
Pre-Training Data	BooksCorpus + English Wikipedia = 16 GB	BERT + CCNews + OpenWebText + Stories = 160 GB	BooksCorpus + English Wikipedia = 16 GB	BooksCorpus + English Wikipedia = 16 GB
Method	Bidirectional Trans-former, MLM & NSP	BERT without NSP, Using Dynamic Masking	BERT Distillation	BERT with reduced para-meters & SOP (not NSP)

Table 2.1: comparing BERT's characteristics with its variants adapted from [29]

The first model we picked was BERT-Base-768-uncased [30] due to its widespread usage and popularity within the deep learning community and the fact that it is the building block for numerous variants. This particular model consists of 12 stacks of encoders with a hidden size of 768.

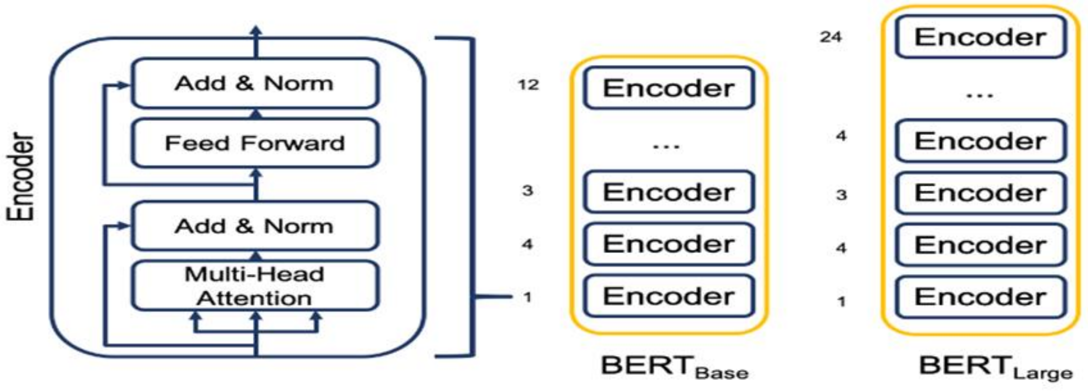


Figure 2.7: BERT encoder stack architecture adapted from [31]

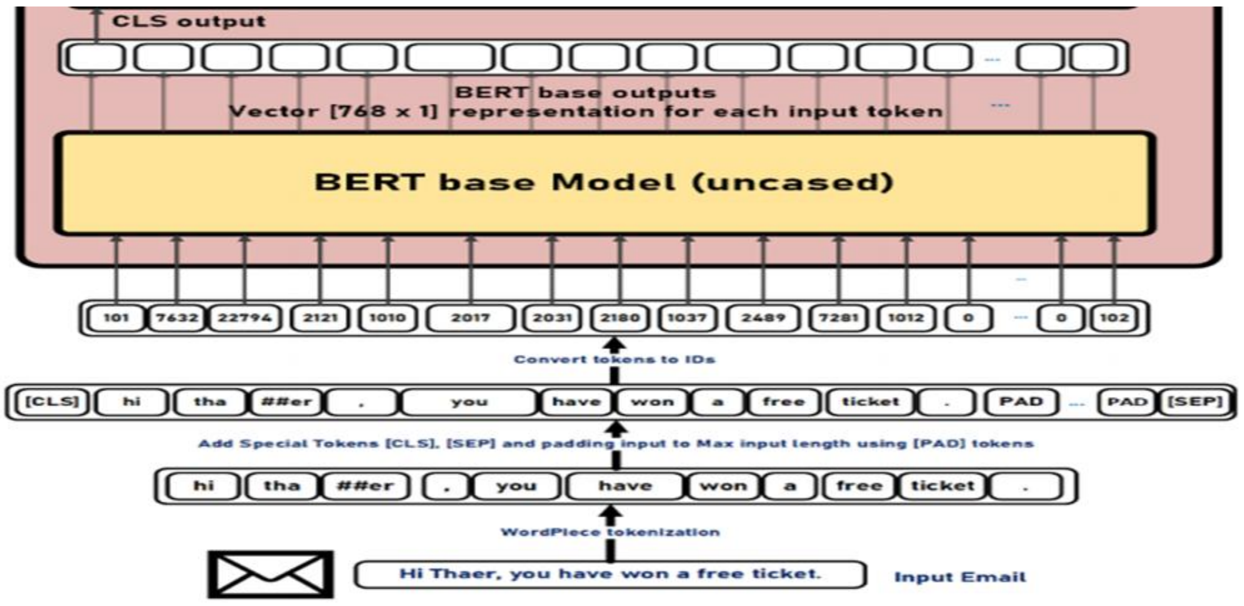

CLS output
BERT base outputs
Vector [768 x 1] representation for each input token
BERT base Model (uncased)
101 7632 22794 2121 1010 2017 2031 2180 1037 2489 7281 1012 0 ... 0 102
Convert tokens to IDs
[CLS] hi tha ##er . you have won a free ticket . PAD ... PAD [SEP]
Add Special Tokens [CLS], [SEP] and padding input to Max input length using [PAD] tokens
hi tha ##er . you have won a free ticket .
WordPiece tokenization
Hi Thaer, you have won a free ticket.
Input Email

Figure 2.8: Taken from [26] with some changes

3.3.1.1 Final layer architecture:

No pre-trained model is good on its own. That's why, according to the learning transfer method, we should incorporate a fine-tuning method that specifies the model for the task of spam detection. This process involves updating the parameters of the pre-trained model, the final layer (classification layer), or both, making the model able to distinguish between spam and ham email effectively.

But before we have to define what a final layer is, a final layer, or classification layer, is a set of neural networks connected straight to the output of BERT's model that does the work of specialization for the task at hand. The number of layers and density of each layer varies.

Our final layer consists of five layers, as shown below, and we'll discuss them layer by layer.

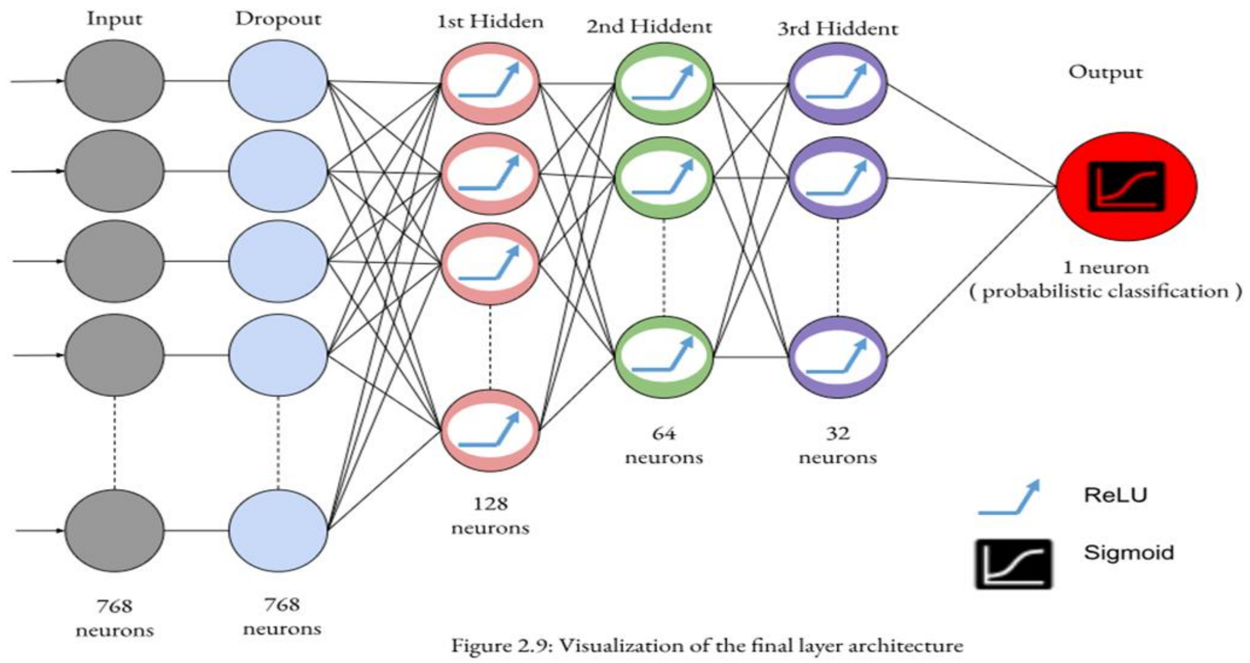
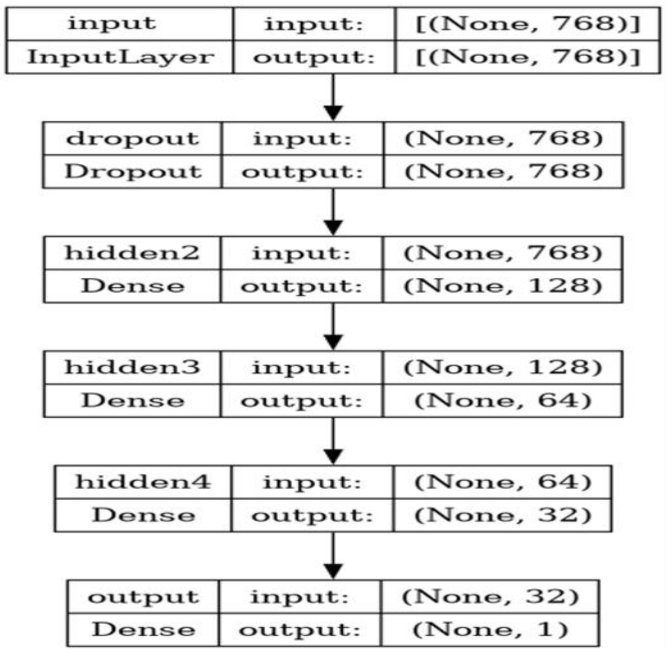


Figure 2.9: Visualization of the final layer architecture



input	input:	[(None, 768)]
InputLayer	output:	[(None, 768)]
dropout	input:	(None, 768)
Dropout	output:	(None, 768)
hidden2	input:	(None, 768)
Dense	output:	(None, 128)
hidden3	input:	(None, 128)
Dense	output:	(None, 64)
hidden4	input:	(None, 64)
Dense	output:	(None, 32)
output	input:	(None, 32)
Dense	output:	(None, 1)

Figure 2.10: Another representation of final layer architecture

A. Dropout:

The dropout layer created by [32], is a regularization technique applied in deep learning that sets the values of neurons to 0 based on a specified percentage chosen by the developer so they can arbitrarily decide the rate at which neurons are excluded. For instance, 0.2 means that 20% of neurons would be forgotten or dropped. As a result, networks are better able to avoid learning unwanted aspects of data, like noise or memorizing the training data, thus ending up with improved results.

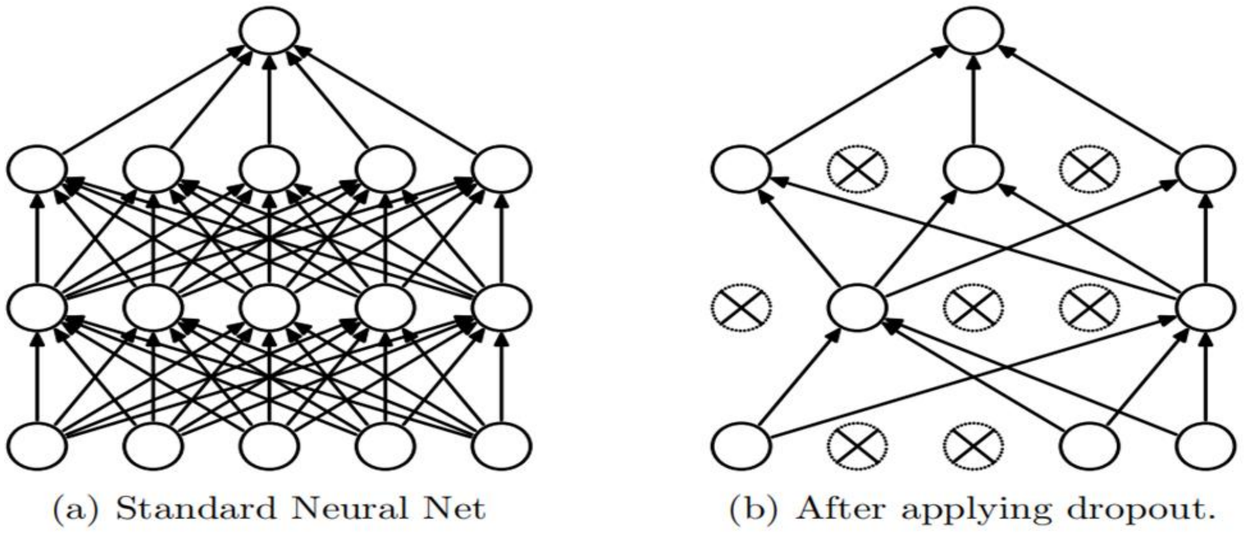


Figure 2.11: Comparison between a standard neural network and after applying dropout layers adopted from [32]

B. Dense Layer

Dense layers consist of fully connected neurons that establish connections with every neuron in the preceding layer. These layers perform transformations followed by an activation function (ReLU in this case) to capture complex patterns in the data.

We chose the hidden neural network sizes of 128, 64, and 32 because it was a good starting point and yielded good results.

3.3.2 DistilBERT:

Every now and then, a new model is added to the BERT family, a bigger and heavier model to improve performance (RoBERTa, for instance). But they are time-consuming and computationally heavy. Sanh et al [18], researched how to improve these aspects of BERT. The result was a distilled version of BERT called DistilBERT.

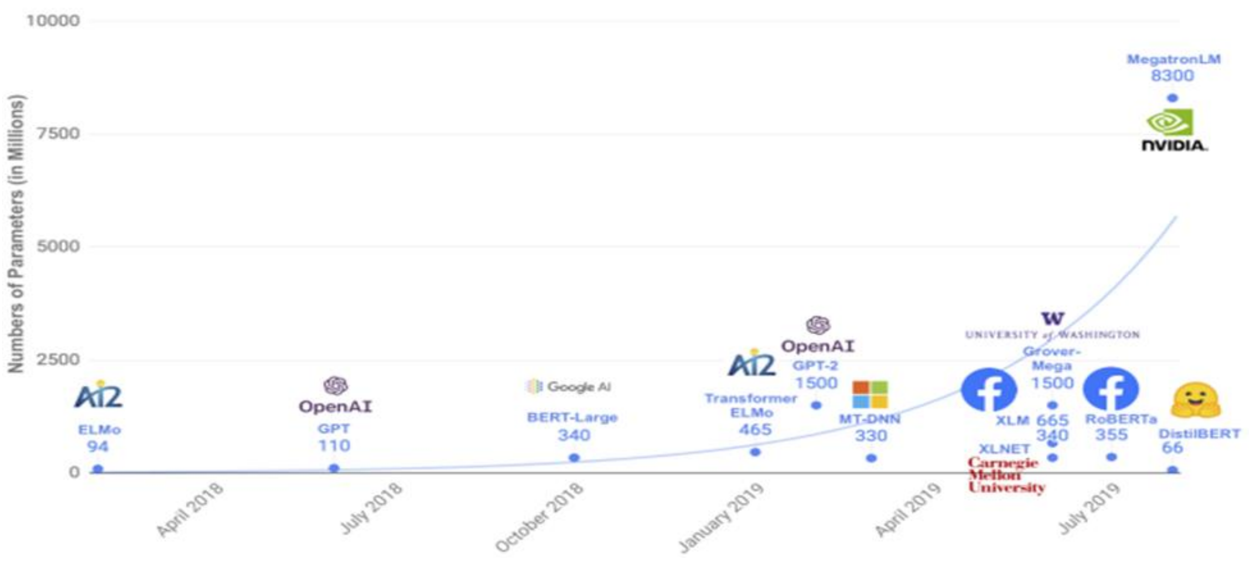


Figure 2.12: Parameter counts of several pre-trained language models adopted from [18]

We chose the distilbert_en_uncased [33], by Tensorflow for the convenience of implementation; this variation comes with 66 million parameters, a stack of 6 transformer encoders with a hidden layer of 786, and 12 attention heads.

We decided on this model because of the enormous savings in computing resources and response times and the smaller memory footprint while maintaining high performance, and it is the ideal model to utilize for experimenting and prototyping our spam detection system. The following figure represents the difference in architecture between BERT and DistilBERT.

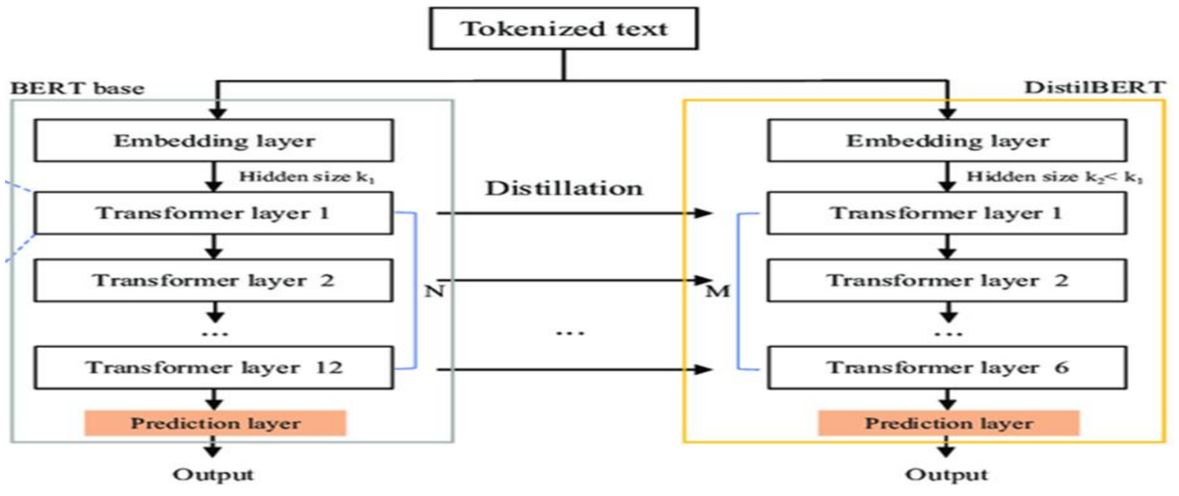


Figure 2.13: The DistilBERT model architecture and components. Adopted from [34]

3.3.2.1 Final layer architecture:

This part discusses the final layer of our second model, which is built using a deeper and more robust network, this network aims to improve performance and provide protections against overfitting. The figure below shows the architecture of the final layer.

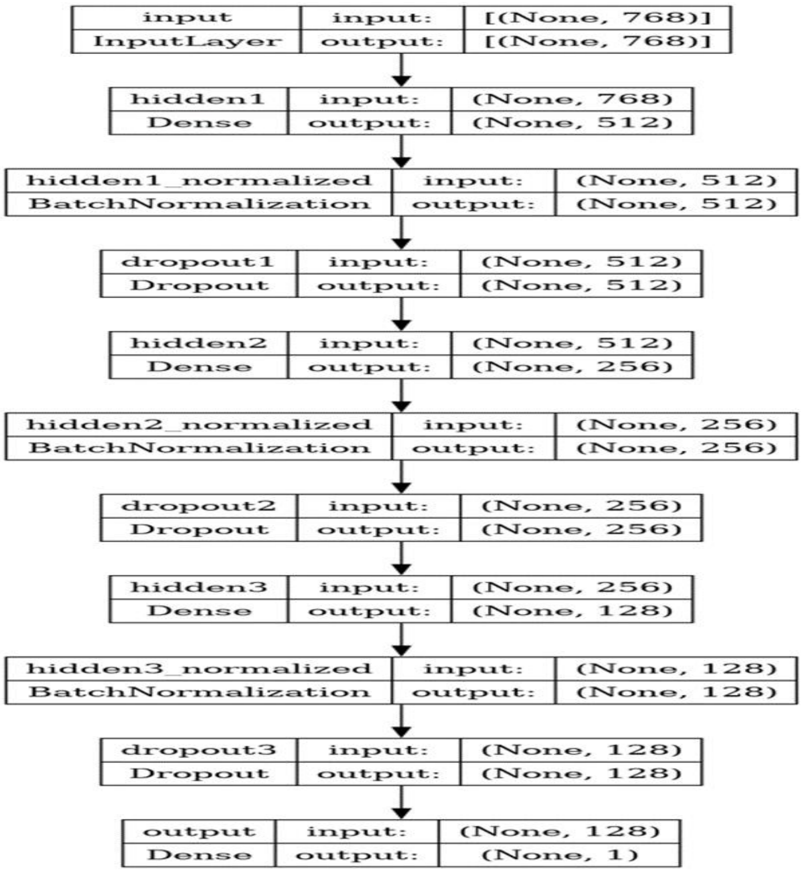

input | InputLayer | input: [(None, 768)] | output: [(None, 768)]
hidden1 | Dense | input: (None, 768) | output: (None, 512)
hidden1_normalized | BatchNormalization | input: (None, 512) | output: (None, 512)
dropout1 | Dropout | input: (None, 512) | output: (None, 512)
hidden2 | Dense | input: (None, 512) | output: (None, 256)
hidden2_normalized | BatchNormalization | input: (None, 256) | output: (None, 256)
dropout2 | Dropout | input: (None, 256) | output: (None, 256)
hidden3 | Dense | input: (None, 256) | output: (None, 128)
hidden3_normalized | BatchNormalization | input: (None, 128) | output: (None, 128)
dropout3 | Dropout | input: (None, 128) | output: (None, 128)
output | Dense | input: (None, 128) | output: (None, 1)

Figure 2.14: DistilBERT final layer architecture

In this revised architecture, several enhancements were introduced to improve the neural network's performance and robustness. As a start, L2 regularization was applied to every dense layer, followed by a batch normalization layer and a dropout.
We enlarged the hidden layer size as followed 512, 256, and 128 neurons, respectively, and then the output layer with 1 neuron. This allows the model to capture more complex patterns within the data.

A. L2 regularization:

L2 regularization [35] (or weight decay) was introduced to all the dense layers by adding a regulation term to the loss function. This discourages large weights in the model, therefore promoting simpler and more generalized solutions. By adding L2 regularization, we aim to mitigate overfitting and improve the model's performance on unseen data. The formulas below show L2 regularization in action.

L2 Regularization

$$\text{Modified loss function} = \text{Loss function} + \lambda \sum_{i=1}^{n} W_i^2$$

Figure 2.15: L2 formula added to the loss fucntion

B. Batch normalization layer:

Following each dense layer, we added layers of batch normalization [36]. Throughout training, these layers change how each layer is activated within small data batches. This change makes training faster and more stable. Another problem the batch normalization tackles is internal covariate shift, which could slow down deep neural network training. Overall, it enhances performance and speeds up the convergence of the model.

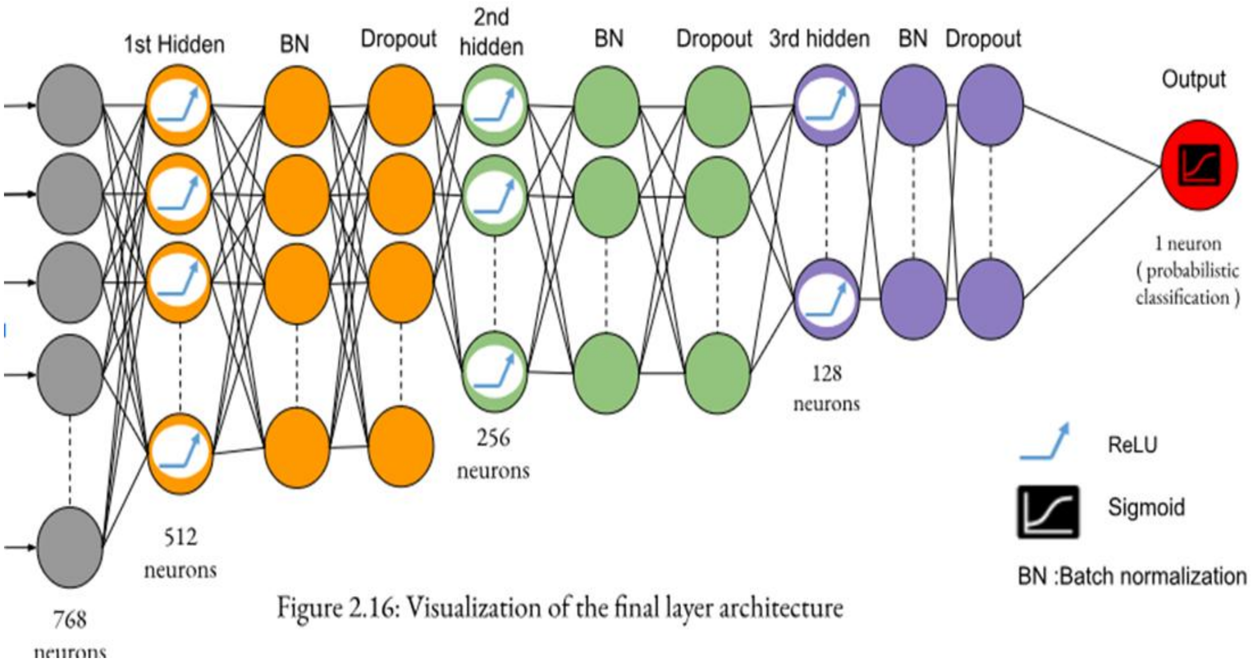


Figure 2.16: Visualization of the final layer architecture

3.3.3 Training

Introduction:

This section discusses the training of the models. This is a crucial stage at which the fine-tuning of the pre-trained representations is done for better suitability to the task of spam detection. We will talk about key points like freezing versus unfreezing of the pre-trained model, hyperparameter tuning, and strategies for mitigating overfitting. Using such methodologies for training, we aim to develop a model for spam detection that is more effective and robust in its operation to help in continued efforts towards digital security.

3.3.3.1 Freezing the pre-trained model;

1 Definition and purpose:

The pre-trained model is frozen when its weights are kept constant throughout the fine-tuning stage. Freezing aims to preserve the knowledge the pre-trained model learned during its first training. And contrary to freezing, unfreezing pre-trained model results in all the weights changing in training.

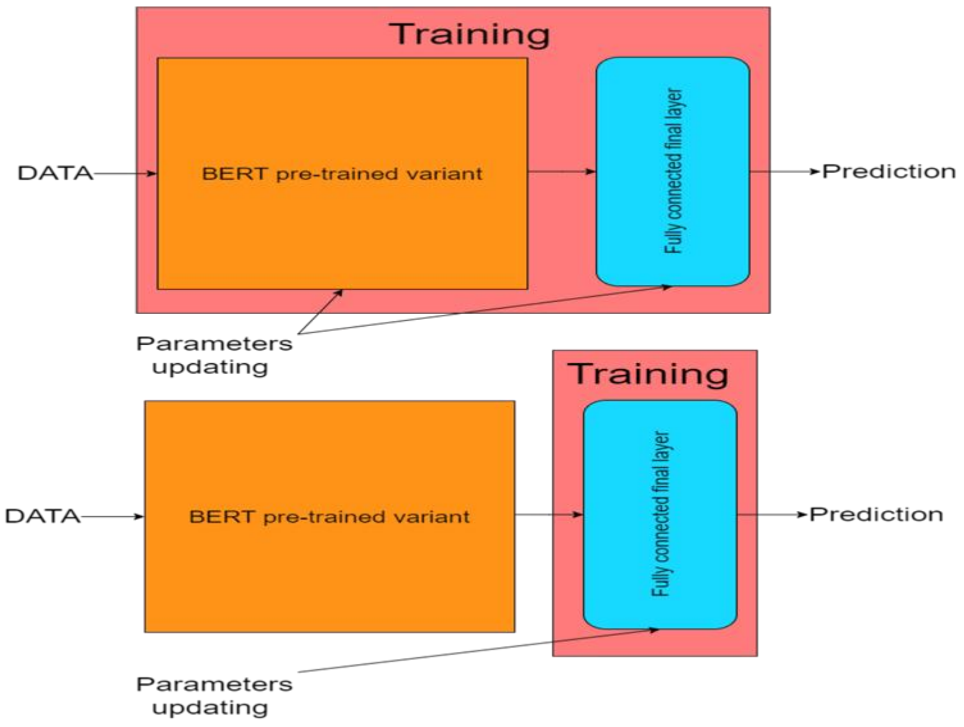


Figure 2.17: Training phase in frozen vs unfrozen pre-trained model

The table below shows key differences between the two approaches

Aspect	Freezing Pre-trained Model	Unfreezing Pre-trained Model
Training Speed	Faster training, as only the final layer is trained	require more time for training as both the pre-trained and additional layers weights are fine-tuned
Computational Resources	Less computational resource are needed	require more computational resources due to training the entire model
Overfitting	Reduced risk of overfitting on small datasets	Increased risk of overfitting due to huge number of parameters
Stability	more stable training process	training process may be less stable due to larger parameter space being fine-tuned
Performance Potential	There is limited potential for performance improvement compared to fine-tuning the entire model	Potential for improved performance by fine-tuning the entire model
Transfer Learning	Limited transferability of knowledge to new tasks due to fixed pre-trained weights	Potential for better transfer learning to new tasks by fine-tuning the entire model
Knowledge forgetfulness	Demonstrates better resistance to forgetting previously learned features due to the preservation of pre-trained layers	Exhibits higher susceptibility to forgetting previously learned features

Table 2.2: comparing key aspects of the frozen and unfrozen pre-trained model

3.3.3.2 Approach selected:

We opted to freeze the pre-trained model for several reasons:

Limited labeled spam datasets

Insufficiently labeled datasets pose a challenge for training a model from scratch.

Complexity of the pre-trained model

LLMs like BERT can reach 340 million parameters making training all of them costly.

Overfitting concerns with limited datasets

With limited labeled data, there's an increased risk of overfitting during model training especially working with such a parameter count (i.e., BERT base 111 million parameters).

Preference for leveraging pre-trained knowledge to expedite training and deployment.

Leveraging knowledge encoded in pre-trained models accelerates the model training process

Reduction of training time and resource requirements

Freezing layers of the pre-trained model reduces the number of trainable parameters, thereby decreasing computational resources and time required for training.

Rapidly evolving spam tactics

The dynamic nature of spam techniques needs models that can adapt quickly to new patterns and variations, making fine-tuning of pre-trained models an appealing choice.

3.3.3.3 Hyperparameter optimization:

We describe in this part the training hyperparameters and the optimization techniques used to improve model performance. Including batch size, regularization techniques, and learning rate, in addition to the reasoning behind the choices.

Value selected			
Hyperparameter	Model 1	Model 2	Reason
Data partition	80% training, 10% validation and 10% testing		allows for effective model training, hyperparameter tuning, and unbiased evaluation of model performance
L2 decay	N/A	0.00001	Balances between regulation against overfitting while not degrading performance
Optimizer	Adam		faster convergence , improved performance and adaptive learning rate
loss	Binary cross entropy		It suits best for binary classification as it measures distance between actual and predicted guiding model toward better classification
Activation function	ReLU		Faster convergence
Final layer architecture	DP/128/64 /32	512/BN/DP/256/B N/DP/128/BN/DP	Discussed earlier
Epoch	30		Balance between over and under fitting
Dropout rate	10%	12%	safeguarding overfitting while maintaining performance
Pre-trained is trainable	False		Faster training ,reduced overfitting risk and less resource intensity

Table 2.3: Showcasing hyperparameters used by each model

3.4 Conclusion:

In conclusion, this chapter emphasizes the importance of informed decisions every step of the way to strike a balance between efficacy and performance, such as model selection, final layer architecture, and hyperparameter optimization. Such decision-making is aimed at providing the best performance of our system in detecting spam messages effectively while remaining mindful of resource constraints.

In the next chapter, we will develop and evaluate two BERT variant models—BERT base and DistilBERT—for spam detection. We will compare their performance, test for overfitting, and benchmark them against related research. The best-performing model will be selected to create a standalone spam testing web application using Django, along with an API for integration into other applications. This approach aims to deliver a robust and accessible spam detection solution.

Chapter Four
Experimentation

Chapter 4 Experimentation

4.1 Introduction:

In today's digital landscape, the never-ending onslaught of spam emails has become an increasingly pervasive problem. Not only do they clutter mail boxes and increase internet traffic, but they also pose serious risks such as fraud and malware infections. In this chapter, we will discuss how we can leverage the state-of-the-art BERT pre-trained models to extract the context of emails and classify them as either spam or ham.

4.2 Description

For this section, we will be validating our spam detection techniques by utilizing two pretrained BERT models on standard datasets. We thoroughly assess the models' performance by conducting extensive testing and comparing them to established benchmarks. Afterwards, we proceed with deploying the models using a website and API, showcasing their real-world usefulness and scalability.

4.3 Datasets

In our research, we opted for the most commonly used datasets in related studies, starting with Enron collection of emails from the Enron Corporation[37], with approximately 30000 emails evenly split into spam and ham and the ling Linguist [38] dataset with almost 3000 email Another dataset we used is SMS spam collection [39], specializing in SMS spam with 5500 emails, and finally the Spamassassin dataset [40] with more than 10,000 emails, making the training datasets for the model ideal as they satisfy criteria like being commonly used in research and verified by experts, We chose to use only the email content for training as it simplifies the workflow and remains focused.

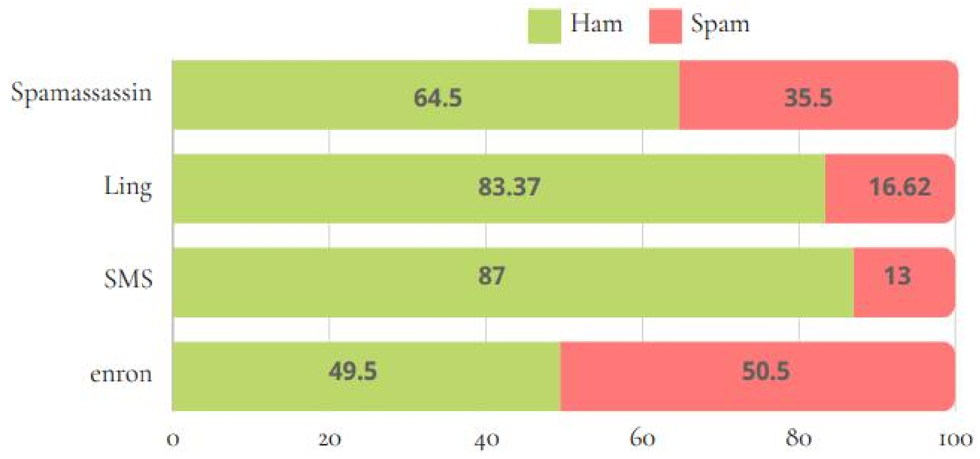


Figure 3.1: shows the ratio of spam and ham in each dataset

4.4 Tools and libraries

4.4.1 Programing language

JavaScript: is a programming language that gives web pages interactive elements that engage a user. It has frameworks that are widely used, namely nodejs and nextjs.

Python: a well-known general purpose programming language mostly used for data science and machine learning.

4.4.2 Libraries:

NumPy: is a Python library used for numerical computation and is essential for data manipulation and analysis.

Pandas: a data analysis library that offers flexible data structures and tools for structured data management and facilitates seamless exploration and manipulation of datasets.

Scikit-learn (Sklearn): free and open-source machine learning library for Python. It offers many tools, like classification regression and clustering algorithms

Seaborn: a statistical data visualization library, helps in the creation of insightful graphs for data analysis and presentation.

Matplotlib: is a general-use visualization library for creating static, animated, and interactive plots, enhancing data visualization capabilities.

4.4.3 Frameworks:

TensorFlow: Developed by Google, TensorFlow is a powerful machine learning framework utilized for building and deploying sophisticated models

Django: is a robust web framework chosen for its scalability .

4.4.4 Workstation

Kaggle (Cloud Spec)

It is a platform for data science competitions and also used for learning, collaboration, and research as it offers great variety of openly public datasets and tutorials and unmatched hardware freely.

4.4.5 Development Tools

- **Pycharm Professional Edition**

4.4.6 Pre-trained Models

1) BERT (Bidirectional Encoder Representations from Transformers)
2) DistilBERT

4.5 Metrics of evaluations

4.5.1 Confusion matrix:

This matrix gives a detailed breakdown of how the model's predictions compare to the actual labels. Correct and incorrect predictions are highlighted by class; this gives insight into model's performance and sheds light on any potential areas for improvement.

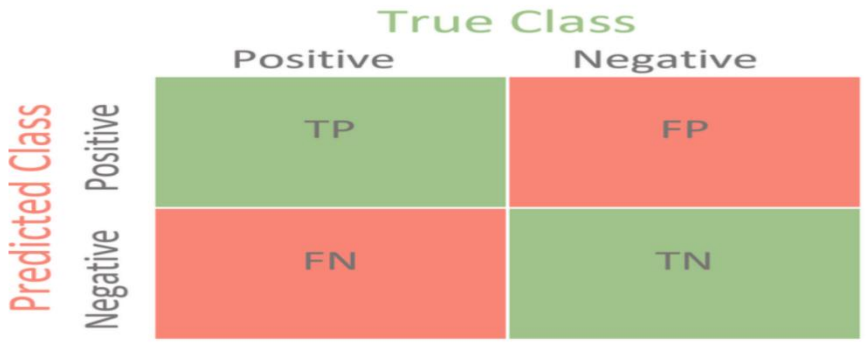


Figure 3.2 : Confusion matrix

True Positive (TP) instances indicate situations in which the model accurately predicted the positive class and the observed value was indeed positive.
False positives (FPs) it's when the model wrongfully predicted the positive class, resulting in a negative actual value.
False negatives (FN) happen when the model made a false prediction about the negative class, resulting in a positive value being predicted.
True Negative (TN): Instances in which the model accurately predicted the negative class and the actual result was negative as well.

Accuracy, precision, and F-measures are then calculated as follows:

4.5.2 Accuracy:

Is the percentage of correct classifications that a trained machine learning model achieves, i.e., the number of correct predictions divided by the total number of predictions across all classes.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN}$$

4.5.2 Precision

Precision quantifies the proportion of true positive predictions out of all positive predictions.

$$Precision = \frac{TP}{TP + FP}$$

4.5.3 Recall

Measures the proportion of actual positive instances that were correctly identified by the model.

$$Recall = \frac{TP}{TP + FN}$$

4.5.4 F-measure

It is mean of accuracy, and recall offers a well-balanced evaluation of the model's performance, which is especially valuable in datasets with unbalanced classes, where one class is more dominant than the other.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall}$$

4.6 Results

This part talks about the results obtained by both models stated in the approach chapter. We will compare them to each other and then to other researchers work and do an overfitting analysis.

The models we opted to use are 2 different models from the same family, Bert-base-uncased-786 and distilbert-base-uncased-786 each with a different final layer architecture and hyperparameters. The table below features key characteristics of each model.

Hyperparameter	Model 1	Model 2
Data partition	80% training, 10% validation and 10% testing	
L2 decay	Not used	0.00001
Final layer architecture	DP/128/64/32	512/BN/DP/256/BN/DP/128/BN/DP
Epoch	30	
Dropout rate	10%	12%
BERT variant	BERT-uncased-base-786	DistilBERT-uncased-base-786

Table 3.1: key difference between the used models

4.6.1 Dataset visualization:

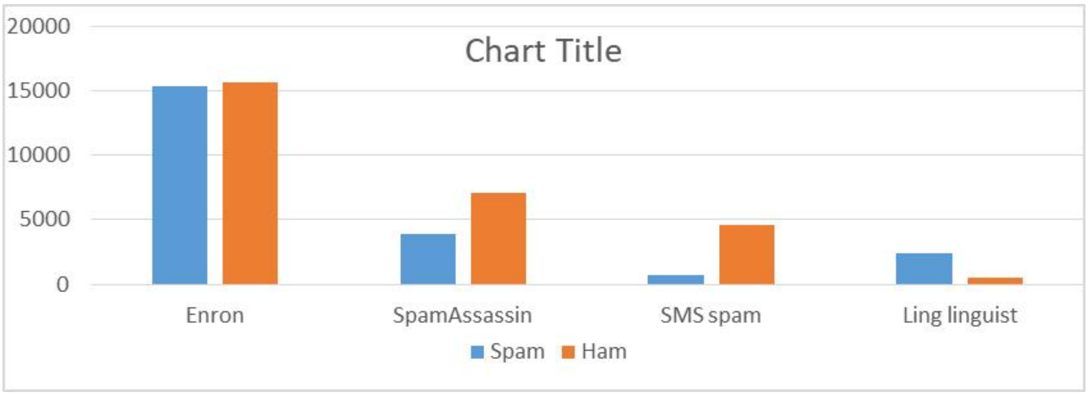
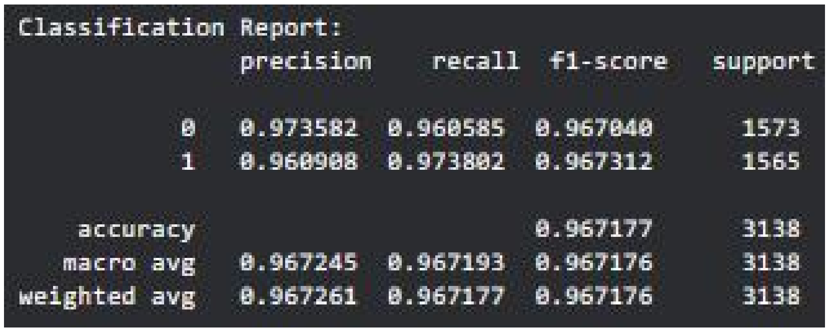


Figure 3.3: Datasets Spam/Ham Ratio

4.6.2 BERT

The first model we will be talking about is the BERT model, with 5 layers in the final layer. We'll illustrate the results in each dataset.

4.6.2.1 Enron



```
Classification Report:
              precision    recall  f1-score   support

           0   0.973582  0.960585  0.967040      1573
           1   0.960908  0.973802  0.967312      1565

    accuracy                       0.967177      3138
   macro avg   0.967245  0.967193  0.967176      3138
weighted avg   0.967261  0.967177  0.967176      3138
```

Figure 3.4: Classification report of the BERT model on Enron dataset

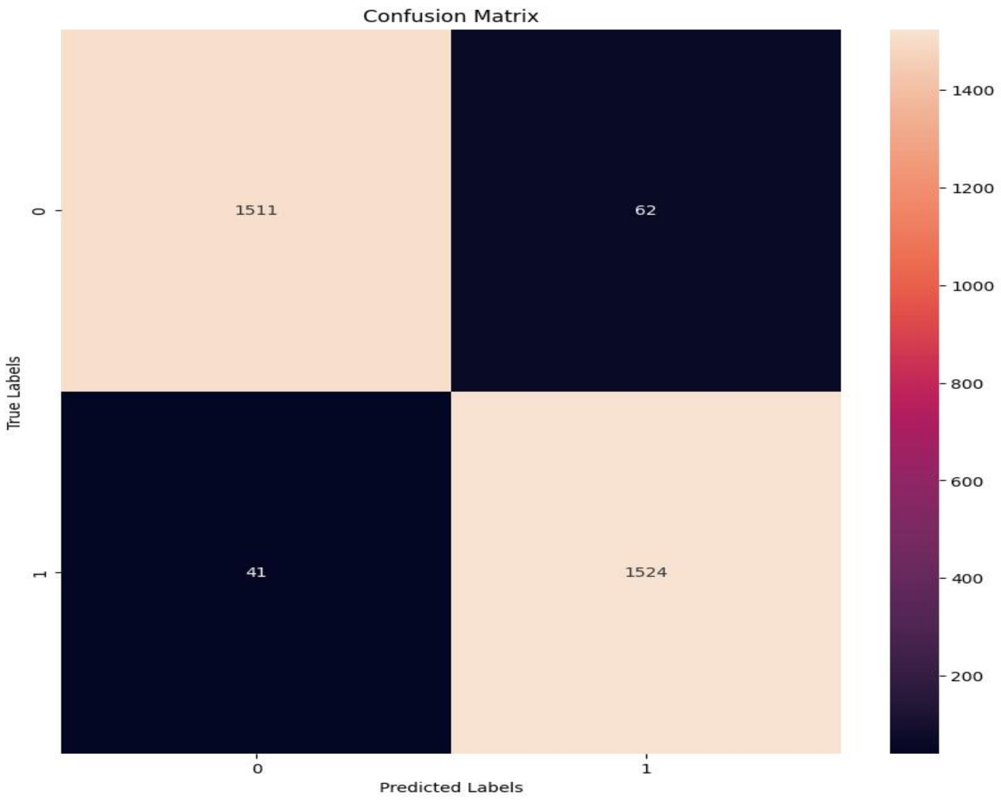
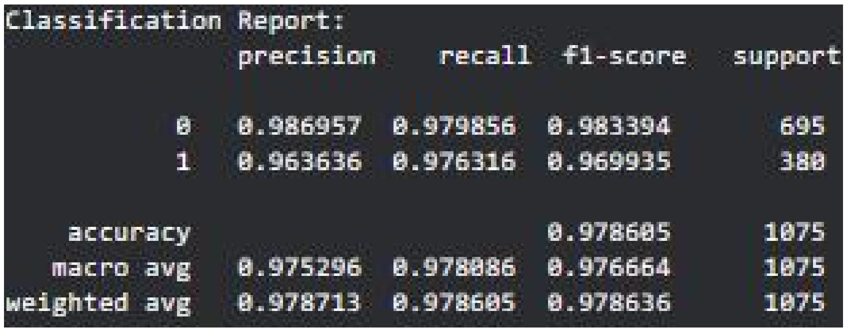


Figure 3.5: Confusion matrix of Enron data on the BERT model

4.6.2.2 SpamAssassin



```
Classification Report:
              precision    recall  f1-score   support

           0   0.986957  0.979856  0.983394       695
           1   0.963636  0.976316  0.969935       380

    accuracy                       0.978605      1075
   macro avg   0.975296  0.978086  0.976664      1075
weighted avg   0.978713  0.978605  0.978636      1075
```

Figure 3.6: Classification report of the BERT model on the Spamassassin dataset

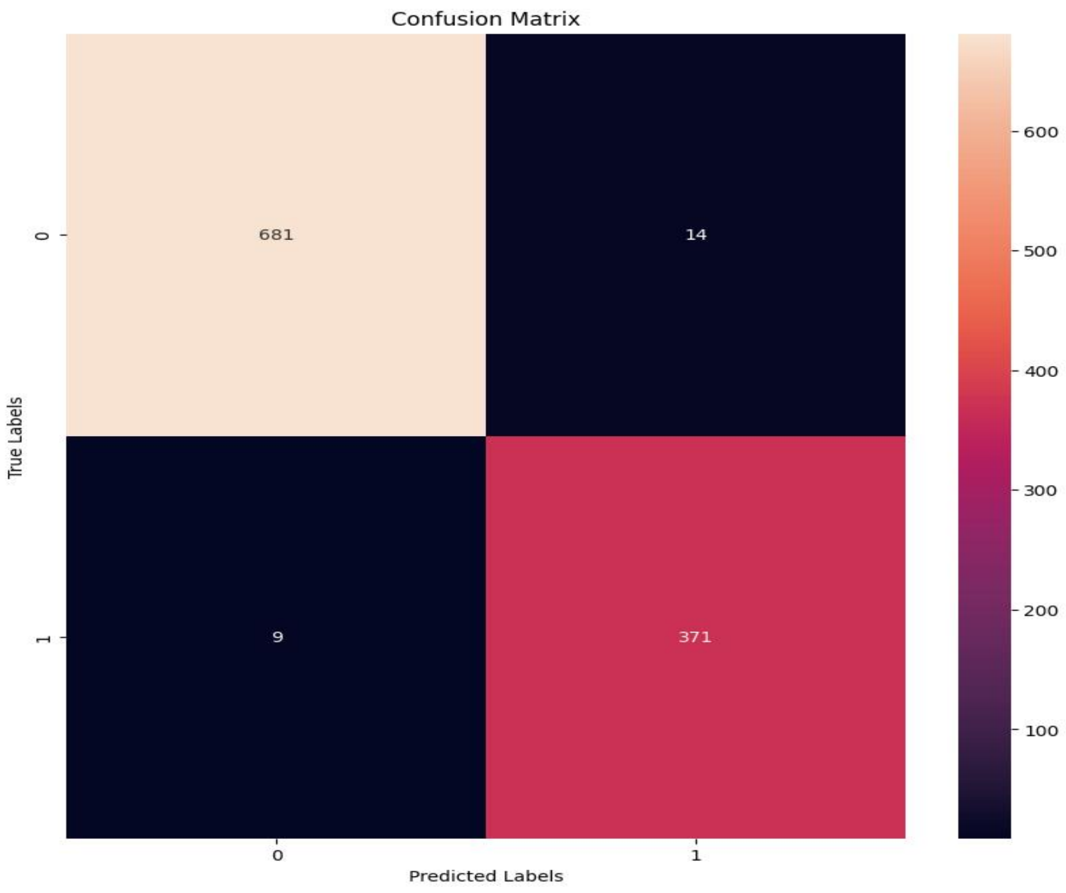
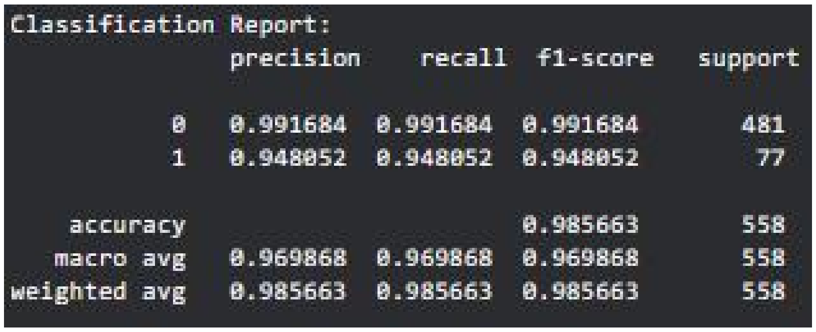


Figure 3.7: Confusion matrix of the BERT model on Spamassassin dataset

4.6.2.3 SMS spam



```
Classification Report:
              precision    recall  f1-score   support

           0   0.991684  0.991684  0.991684       481
           1   0.948052  0.948052  0.948052        77

    accuracy                       0.985663       558
   macro avg   0.969868  0.969868  0.969868       558
weighted avg   0.985663  0.985663  0.985663       558
```

Figure 3.9: Confusion matrix of the BERT model on the SMS spam dataset

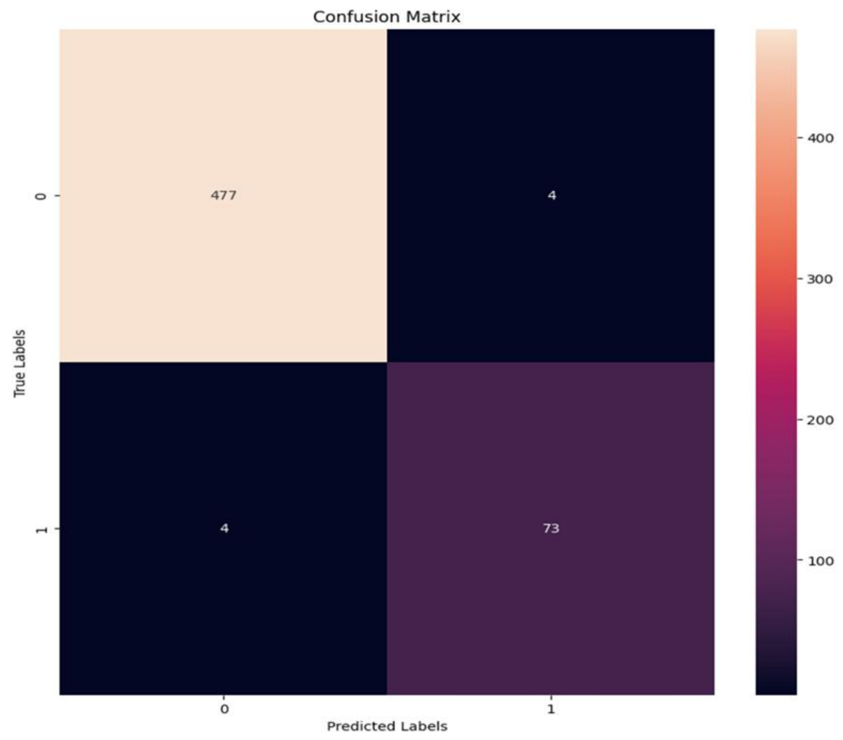
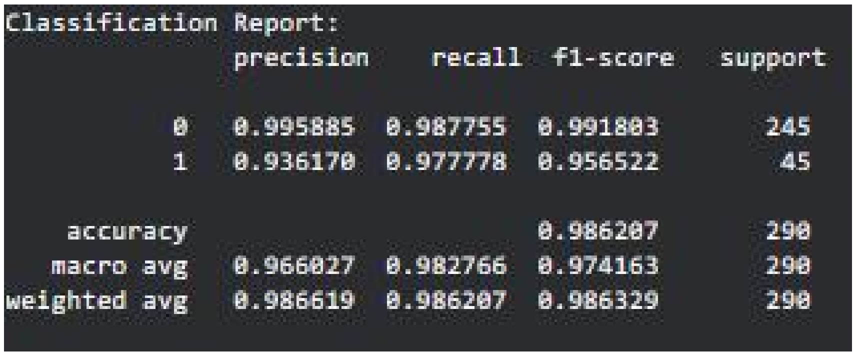


Figure 3.9: Confusion matrix of the BERT model on the SMS spam dataset

4.6.2.4.Ling Linguist



```
Classification Report:
              precision    recall  f1-score   support

           0   0.995885  0.987755  0.991803       245
           1   0.936170  0.977778  0.956522        45

    accuracy                       0.986207       290
   macro avg   0.966027  0.982766  0.974163       290
weighted avg   0.986619  0.986207  0.986329       290
```

Figure 3.10: Classification report of BERT model on Ling spam dataset

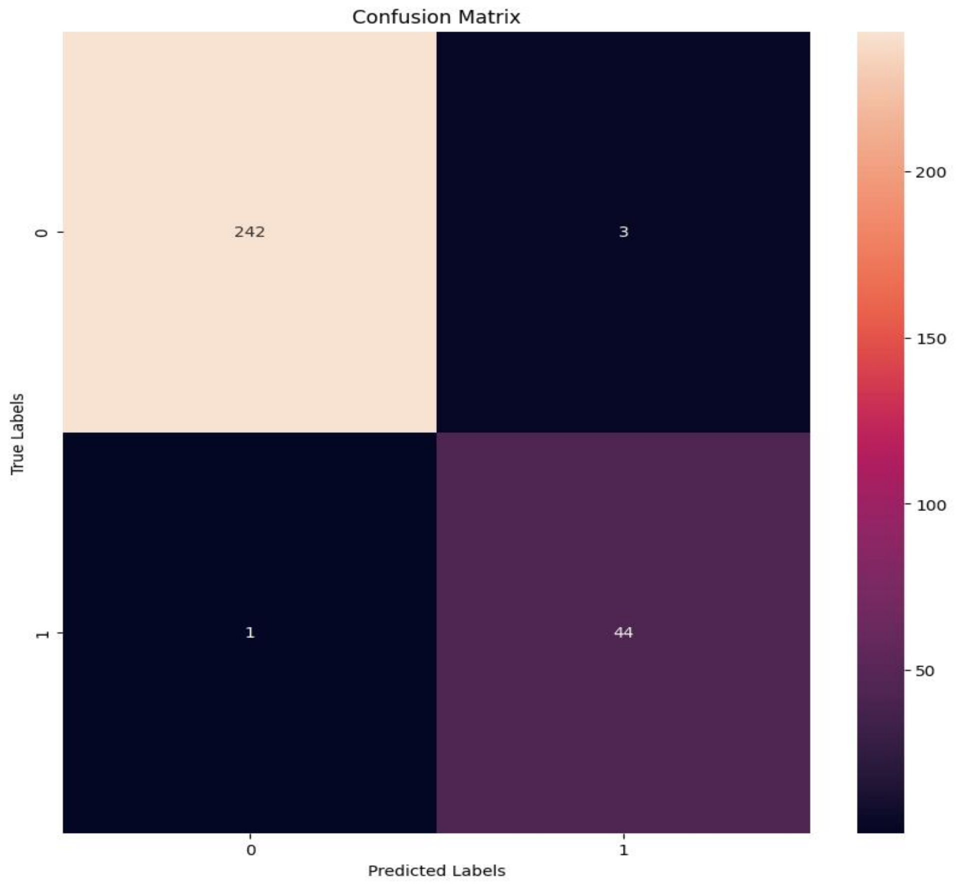
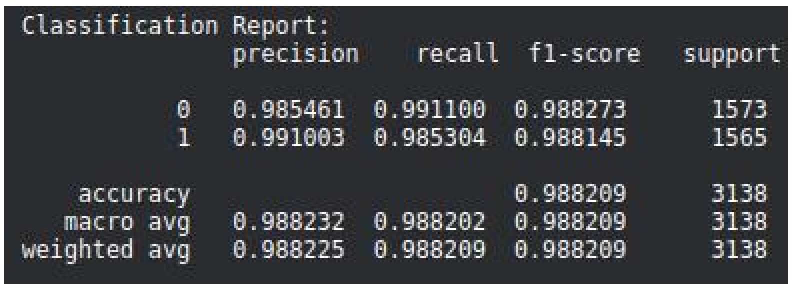


Figure 3.11: Confusion matrix of the BERT model on the Ling dataset

4.6.3 DistilBERT

Our second model is lightweight and reinforced with a deeper architecture. The results are as follows:

4.6.3.1 Enron



```
Classification Report:
              precision    recall  f1-score   support

           0   0.985461  0.991100  0.988273      1573
           1   0.991003  0.985304  0.988145      1565

    accuracy                       0.988209      3138
   macro avg   0.988232  0.988202  0.988209      3138
weighted avg   0.988225  0.988209  0.988209      3138
```

Figure 3.12: Classification report of DistilBERT on the Enron dataset

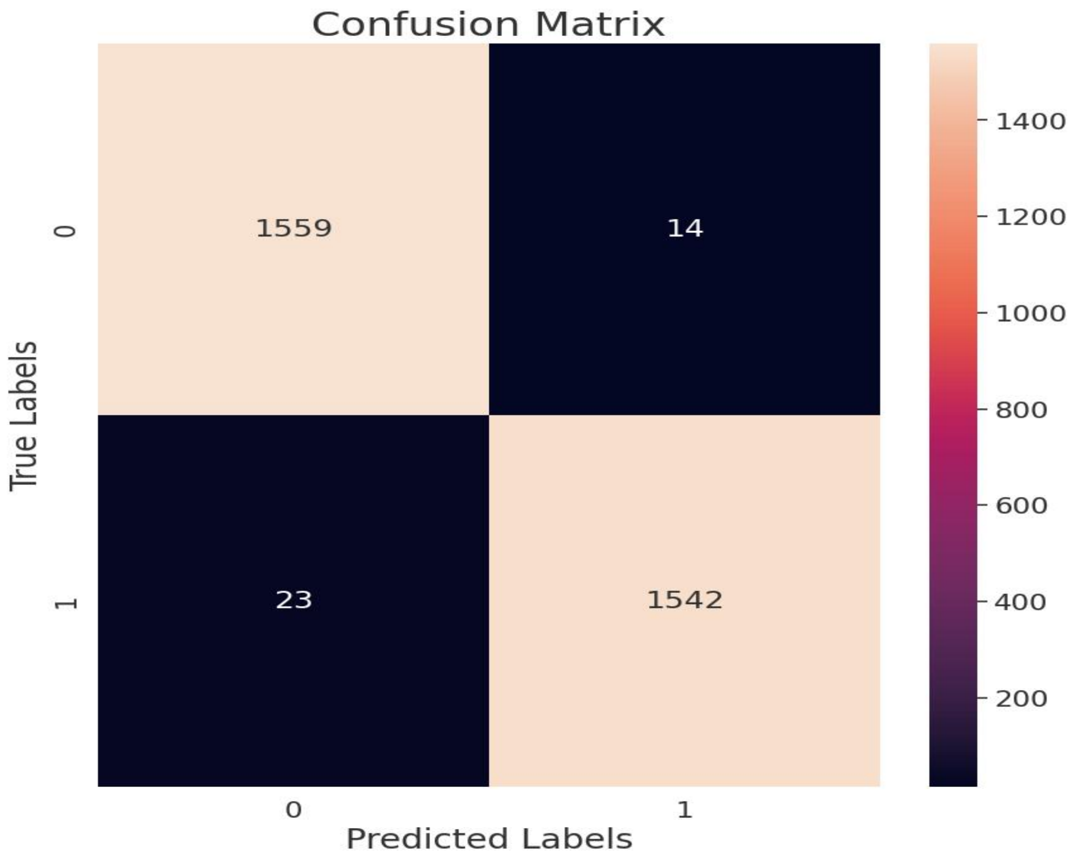
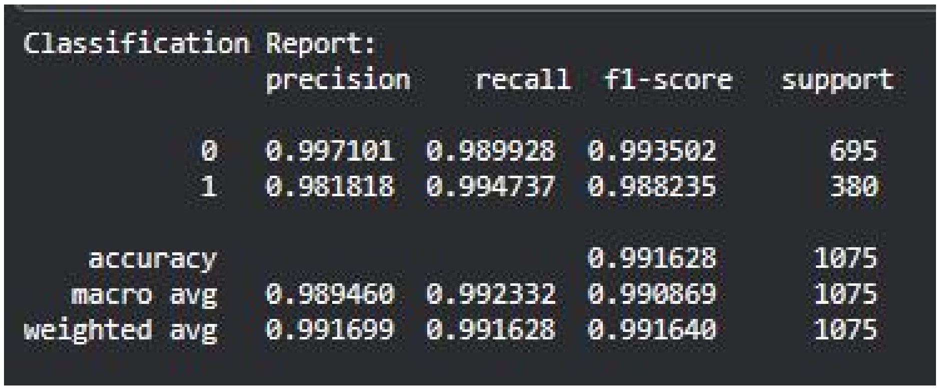


Figure 3.13: Confusion matrix of the DistilBERT on the Enron

4.6.3.2 SpamAssassin



```
Classification Report:
              precision    recall  f1-score   support

           0   0.997101  0.989928  0.993502       695
           1   0.981818  0.994737  0.988235       380

    accuracy                       0.991628      1075
   macro avg   0.989460  0.992332  0.990869      1075
weighted avg   0.991699  0.991628  0.991640      1075
```

Figure 3.14: Classification report of the DistilBERT on the Spamassassin dataset

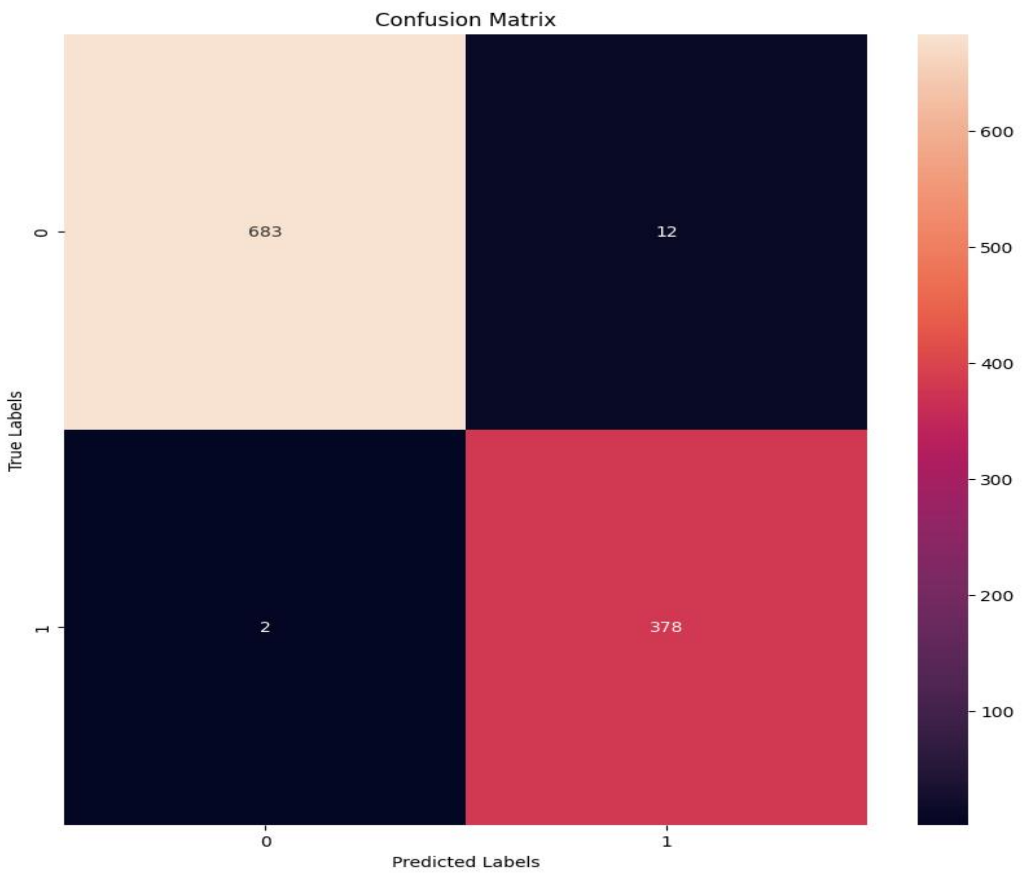
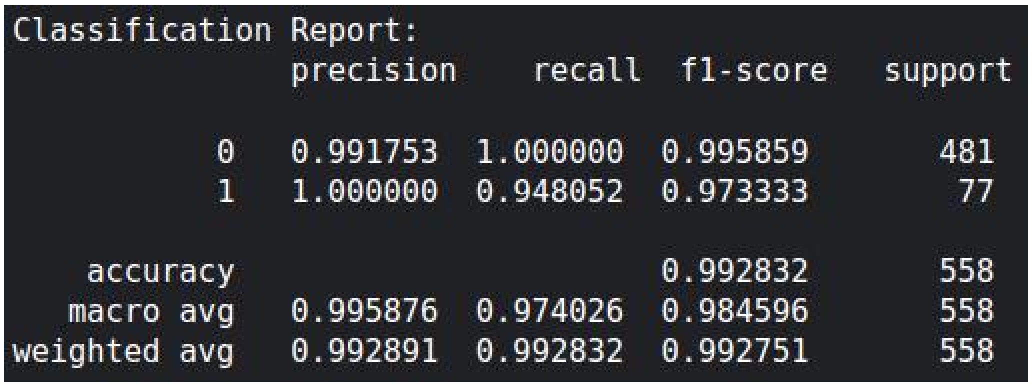


Figure 3.15: Confusion Matrix of the DistilBERT on Spamassassin dataset

4.6.3.3 SMS spam



```
Classification Report:
              precision    recall  f1-score   support

           0   0.991753  1.000000  0.995859       481
           1   1.000000  0.948052  0.973333        77

    accuracy                       0.992832       558
   macro avg   0.995876  0.974026  0.984596       558
weighted avg   0.992891  0.992832  0.992751       558
```

Figure 3.6: Classification report of the DistilBERT model on the SMS dataset

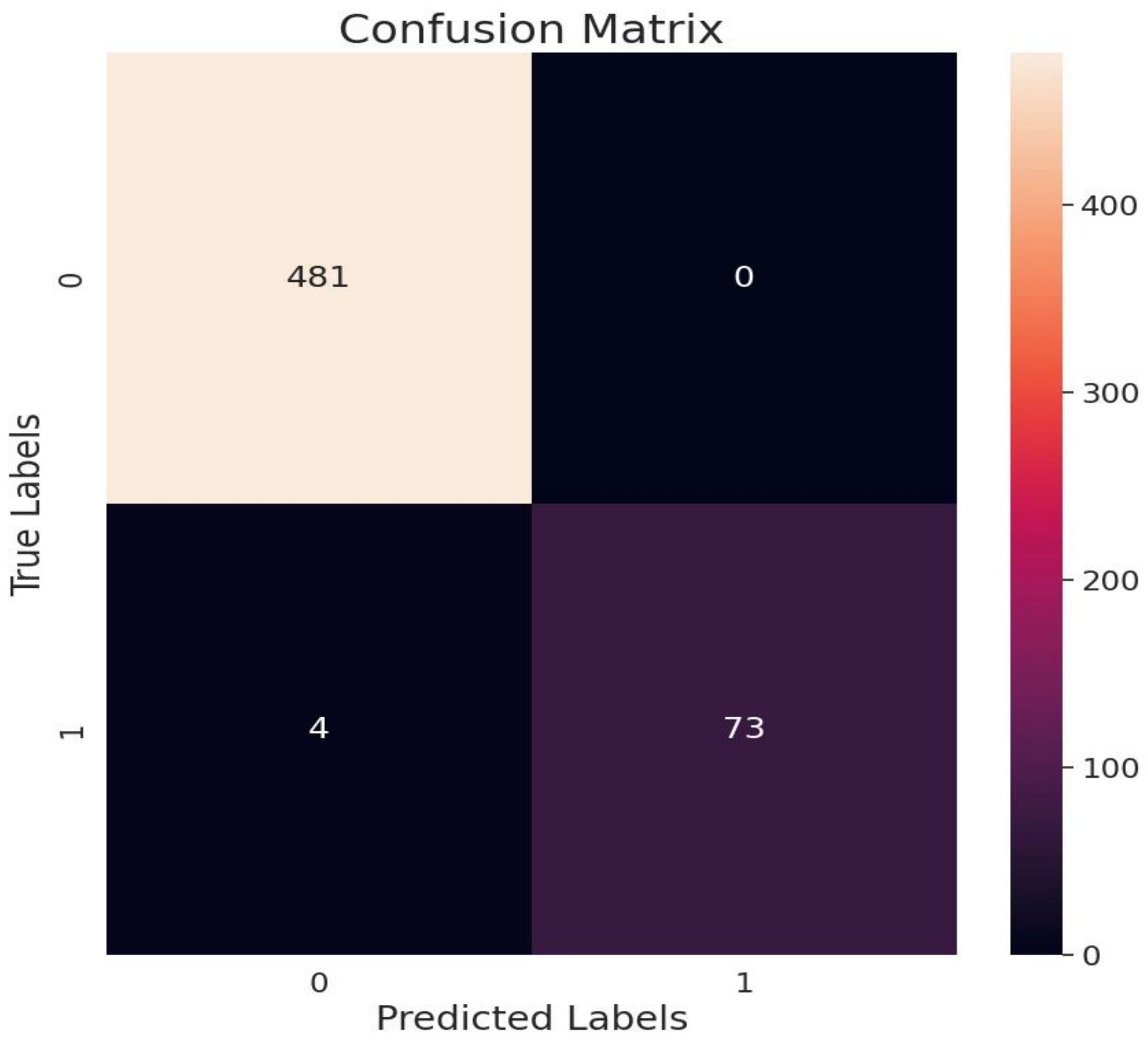
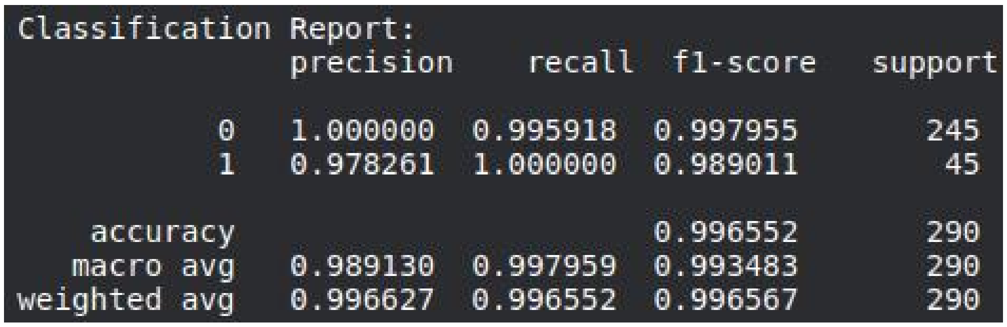


Figure 3.17: Confusion matrix of the DistilBERT on the SMS spam dataset

4.6.3.4 Ling Linguist dataset



```
Classification Report:
              precision    recall  f1-score   support

           0   1.000000  0.995918  0.997955       245
           1   0.978261  1.000000  0.989011        45

    accuracy                       0.996552       290
   macro avg   0.989130  0.997959  0.993483       290
weighted avg   0.996627  0.996552  0.996567       290
```

Figure 3.18: Classification report of the DistilBERT on Ling spam dataset

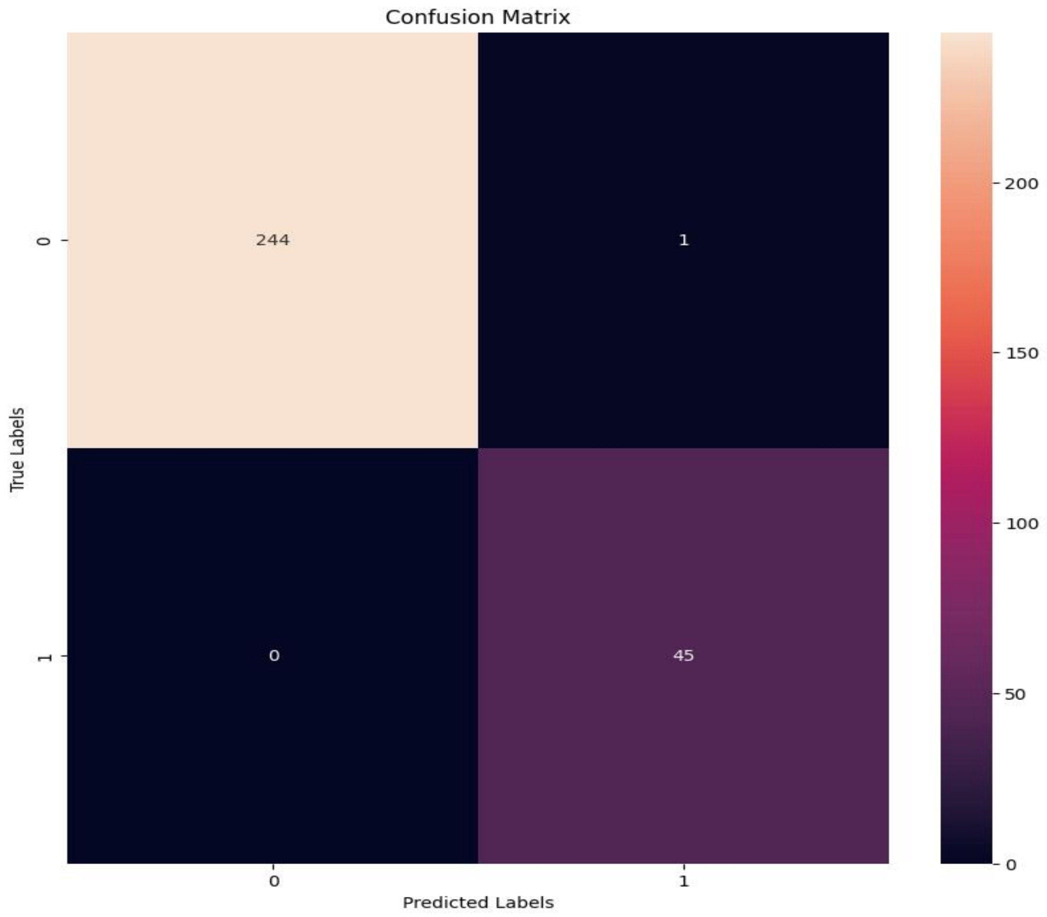


Figure 3.19: Confusion matrix of the DistilBERT model on Ling spam dataset

The table below summarizes the results and compares the two models.

	BERT model		DistilBERT	
Dataset	F1-score	accuracy	F1-score	accuracy
Enron	96,7040%	96.7177%	98,8273%	98.8273%
Spamassassin	98,3394%	97.8605%	99,3502%	99.1628%
SMS spam	99,1684%	98.5663%	99.5859%	99.2832%
Ling linguist	99,1803%	98.6207%	99,7955%	99.6552%

Table 3.2: Table comparing BERT and DistilBERT performance on different datasets

Authors	Year	Datasets	Performance	
			F1-score	Accuracy
Thaer Sahmoud, Dr. Mohammad Mikki [26]	2022	Enron Spam Dataset Spamassassin Ling-Spam Dataset SMS Spam Collection	98,62 % 97,83 % 99,13 % 99,28 %	Not mentioned
VS Tida, S Hsu [28]	2022	Spamassassin Enron Spam Dataset Spam Text Ling-Spam Dataset	98 % 97 % 98 % 98 %	97 % 97 % 93 % 94 %
C Oswald, SE Simon, A Bhattacharya [27]	2022	Enron Spam Dataset SMS Spam Collection Spamassassin Ling-Spam Dataset	97.23 % 98.07 % 97.89 % 98.45 %	97 % 97 % 97 % 98 %
Presented Work	2024	Enron Spam Dataset Spamassassin SMS Spam Collection Ling spam dataset	**98,8273 %** **99,3502 %** **99.5859 %** **99,7955 %**	**98.8273 %** **99.1628 %** **99.2832 %** **99.6552 %**

Table 3.3: Summary of work that has used the BERT approach

4.7 Discussion

4.7.1 Overfitting analysis

4.7.1.1 Introduction to Overfitting

Important concepts in machine learning, overfitting and underfitting are particularly applicable to spam detection using BERT models. When a model learns the training data too well, it overfits, capturing noise and not being able to generalize to new, unknown data. Contrary, underfitting happens when a model is too basic to represent the underlying structure of the data, leading to poor performance even on the training set.

The phenomenon of overfitting and underfitting in the overall context of a BERT-based spam detection system is the main subject in this section. Our goal is to maximize model performance and ensure trustworthy spam detection abilities by knowing the symptoms and consequences of overfitting and underfitting

4.7.1.2 Overfitting

In overfitting, a model becomes so good at our training data that it has mastered every pattern, including noise. This makes the model perform well with training data but poorly with test or validation data [41].

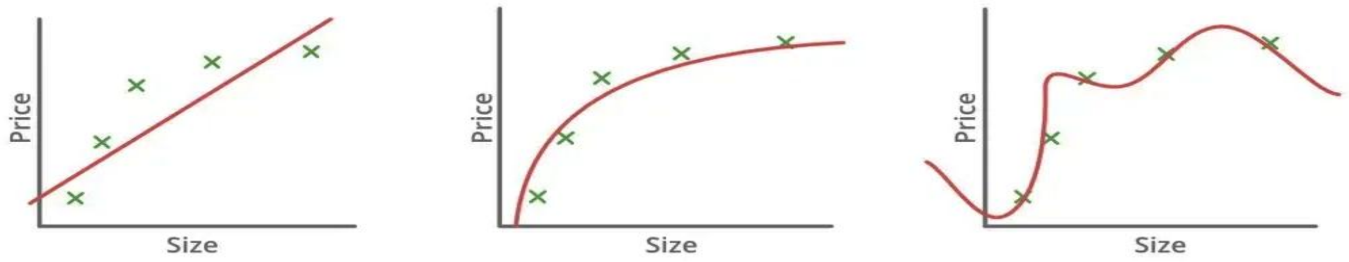


Figure 3.20: How an underfitted, optimal and overfitted model fits data, adopted from [42]

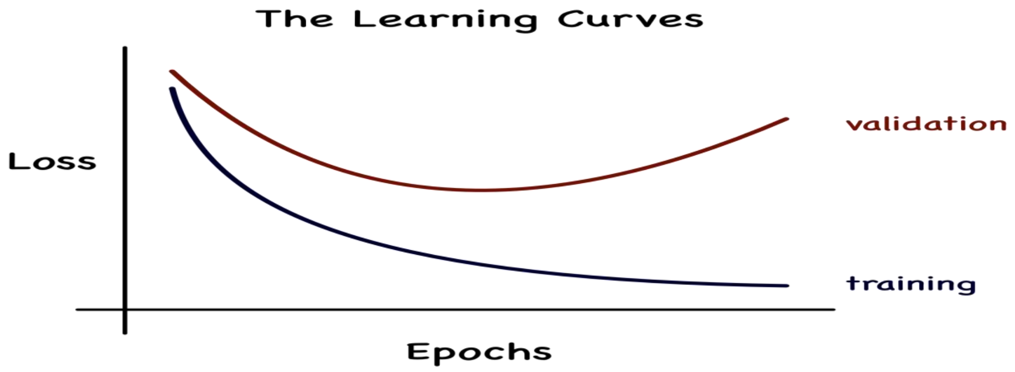


Figure 3.21: loss vs. epoch graph of an overfitted model

Solution:

- Simplify the model
- Incorporate regulation (L1, L2, dropout, etc.)
- Early stopping

4.7.1.3 Underfitting

Underfitting occurs when a model is too simple to capture patterns in the dataset provided, and that causes the model to perform poorly on both the training and test data.

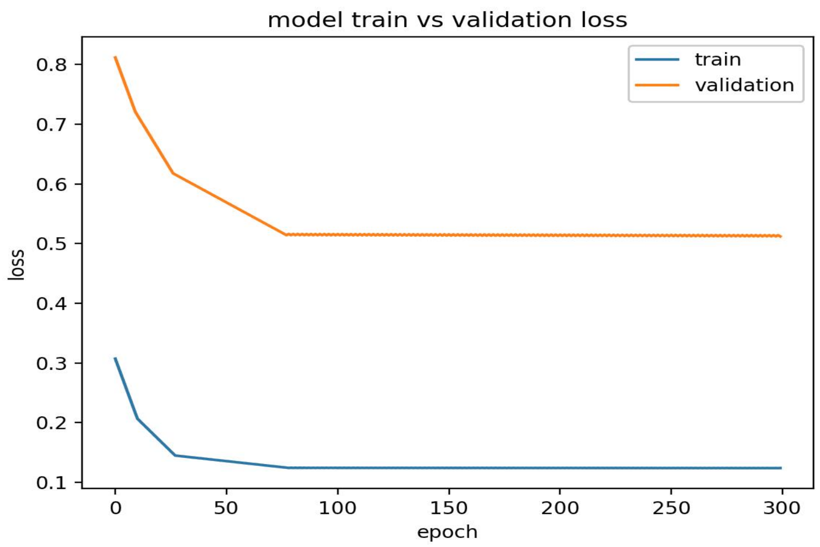


Figure 2.22: loss / epoch graph showing an underfitted model adapted from [43]

Solution:

- Increase model complexity
- Reduce regulation strength (L1, L2, dropout, etc.)
- Hyperparamter tuning

4.7.1.4 Model analysis

How to know if a model is overfitting or underfitting ?

- **Model accuracy:**

  A high training accuracy but low validation accuracy indicates overfitting. Low accuracy in both training and validation indicates underfitting.
- **Class-wise measures**
  - **F1-score**

    Indication:

    Large discrepancies between class-wise training and validation F1-scores can signal overfitting. Consistently low F1-scores for certain classes on both training and validation sets can indicate underfitting.
  - **Recall**

    Indication:

    High recall in training and low recall in validation indicate overfitting.
    Low recall on both can indicate underfitting.
  - **Precision**

    Indication:

    High precision in training but low precision in validation indicates overfitting. Low precision in both can indicate underfitting.

4.7.2 BERT base model

4.7.2.1 Enron dataset

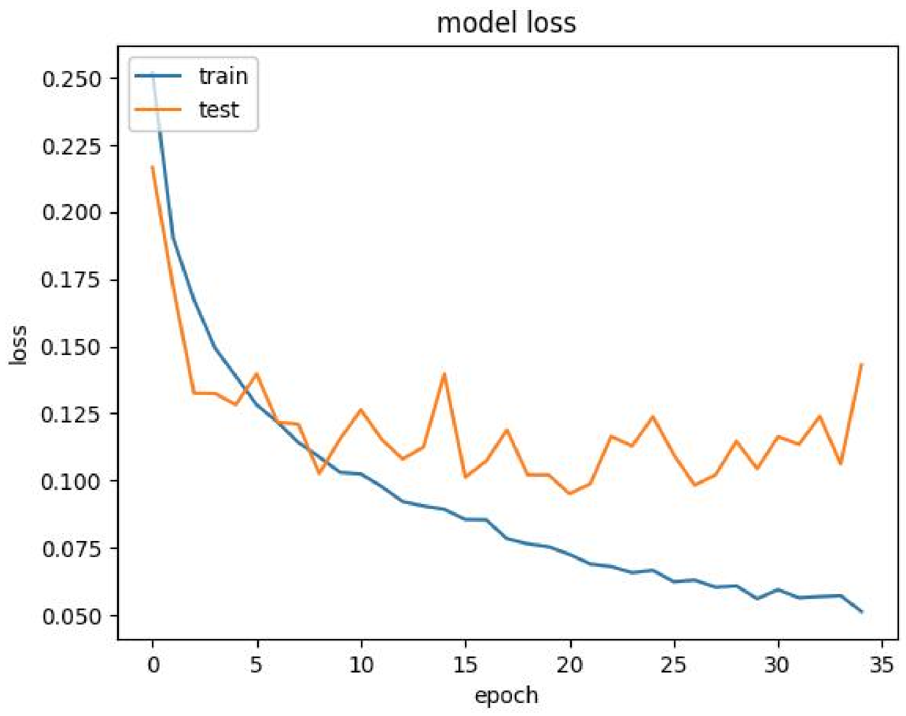


Figure 3.23: train and validation loss against epoch count

4.7.2.2 Spamassassin dataset

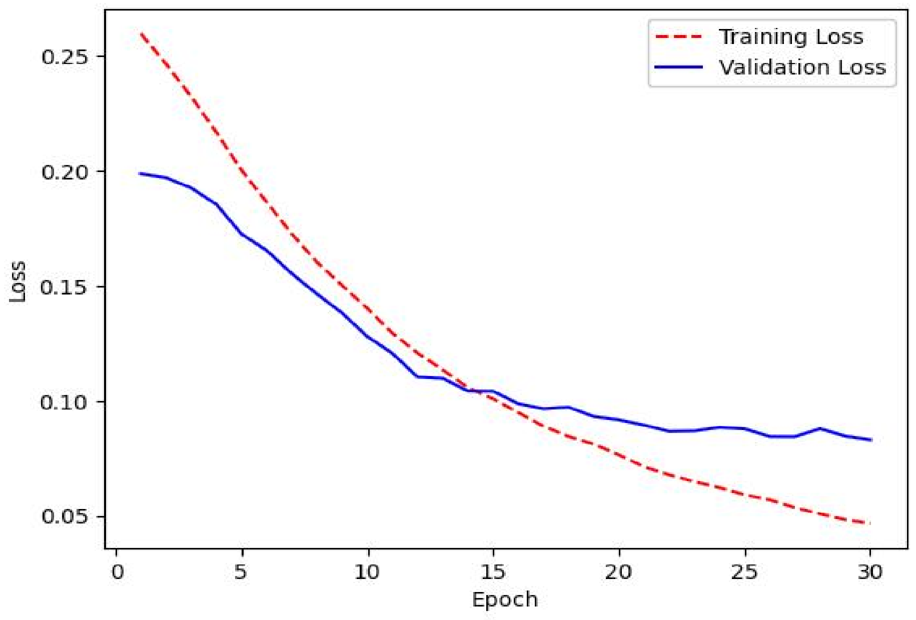


Figure 3.24: train and validation loss against epoch count

4.7.2.3 SMS spam dataset

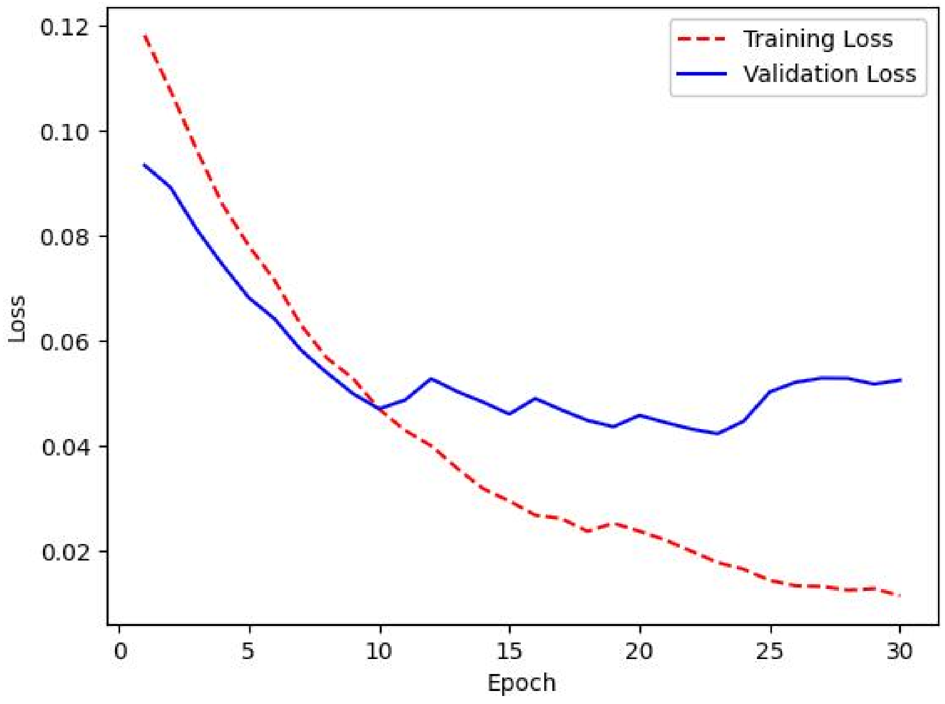


Figure 3.25: train and validation loss against epoch count

4.7.2.4 Ling Linguist dataset

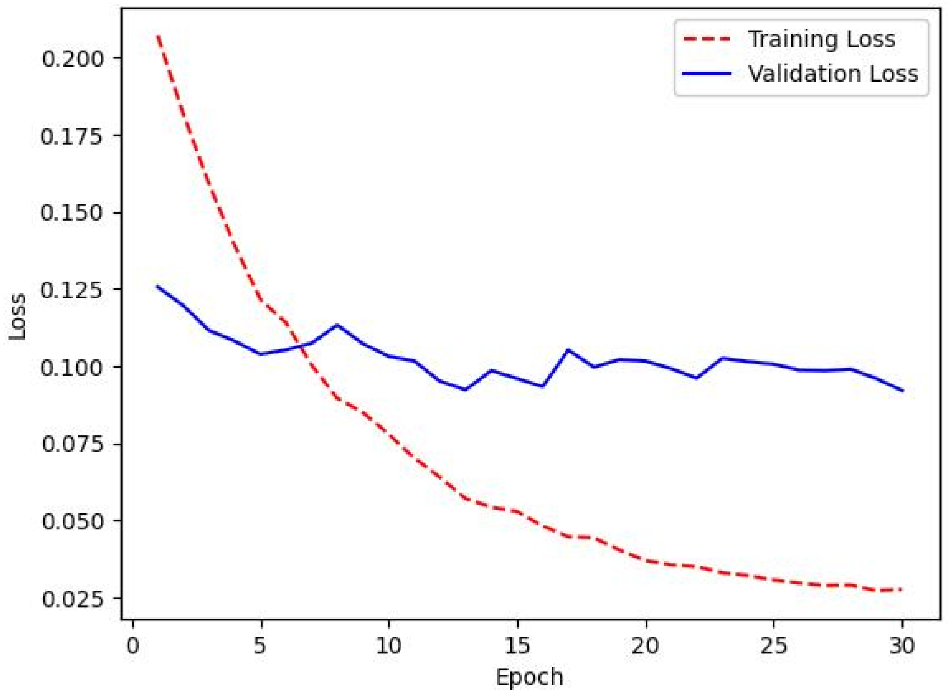


Figure 3.26: train and validation loss against epoch count

4.7.2.5 Interpretation of results:

By viewing the classification reports of our BERT model and the train and validation loss against epoch, we can determine several key characteristics of the model's behavior

		Enron	Spamassassin	SMS spam	Ling linguist
accuracy		96.7177%	97.8605%	98.5663%	98.6207%
F1-score		96,7040%	98,3394%	99,1684%	99,1803%
Precision	0	97.3582%	98.6957%	99.1684%	99.5885%
	1	96.0908%	96.3636%	94.8052%	93.6170%
recall	0	96,0585%	97,9856%	99,1684%	98,7755%
	1	97,3802%	97,316%	94,8052%	97,7778%

Table 3.4: resulted metrics for the BERT model.

- **Model accuracy:**

The BERT model with high overall accuracy demonstrates good performance on all the datasets; however, accuracy by itself can be deceiving, especially if the dataset is biased. Other metrics like F1-score, precision and recall help us understand our model better.

- **Class wise measures**
  - **f1-score**

All of the classes in the dataset have high F1-scores which indicate effective classification with minimal false negatives and false positives.

  - **Recall**

On both classes, the model performed well with small difference indicating the model's ability to capture all the instances without bias

  - **Precision**

A relatively higher precision rates for both classes indicates correct predictions in most cases. The precision for minority class is slightly lower yet very high indicating a reliable procedure for detecting spams that has few false positives.

- **Overfitting or underfitting indicators:**

Closing the gap between training and validation loss:

Significance:

When validation loss starts plateauing, it means that the model has learned as much as possible from the training data without overfitting.

Implication: In our case, validation loss shows that this is where our model reached an optimal generalization level beyond which further training wouldn't significantly improve performance.

Validation Loss Plateauing:
Significance: as the validation loss stops decreasing, this means that the model learned as much as it could from the train data without overfitting.
Implication: The validation loss in our case means that the model reached an optimal generalization level where further training would not significantly improve its performance on the validation set.

Very Good Model Results:
Significance: high accuracy, high precision, good recall, and F1-score, meaning that the model makes accurate predictions in both training and validation sets.
Implication: The model's performance is strong; therefore, meaningful patterns have been established based on training samples, and these can be successfully applied to unseen data.

Conclusion:

The convergence of both training and validation losses, along with good outcomes, confirms the model's ability to generalize well to unseen data. Therefore indicating the readiness of the model.

4.7.3 DistilBERT model

4.7.3.1 Enron

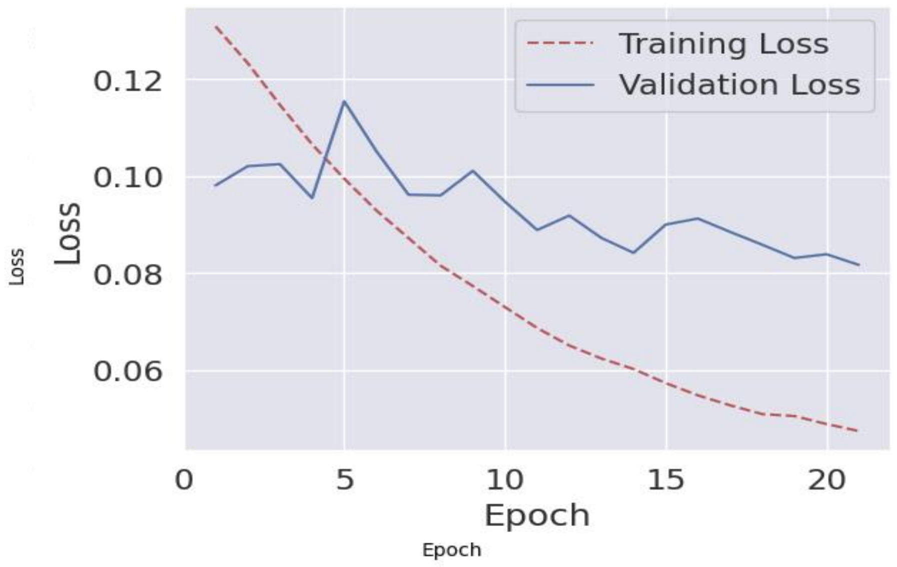


Figure 3.27: train and validation loss against epoch count

4.7.3.2 Spamassassin

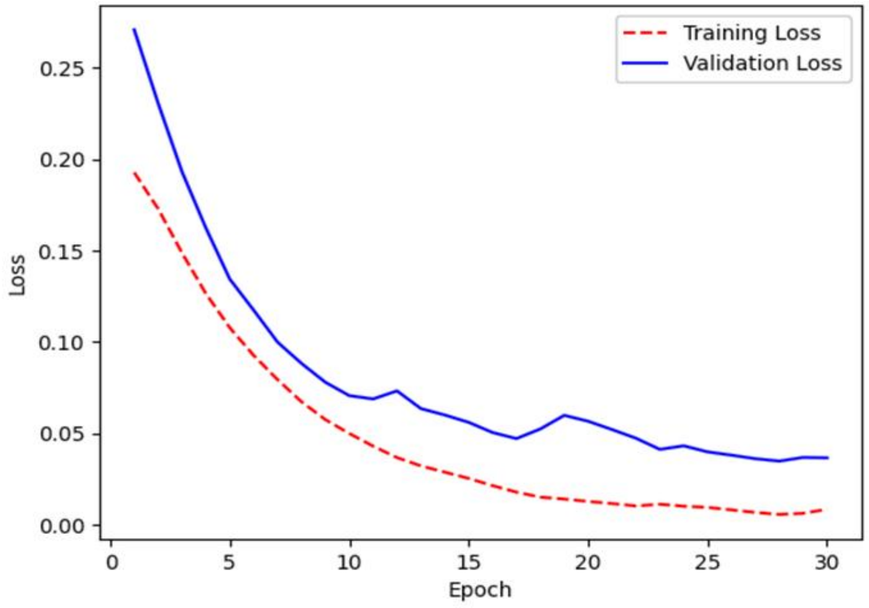


Figure 3.28: train and validation loss against epoch count

4.7.3.3 SMS spam

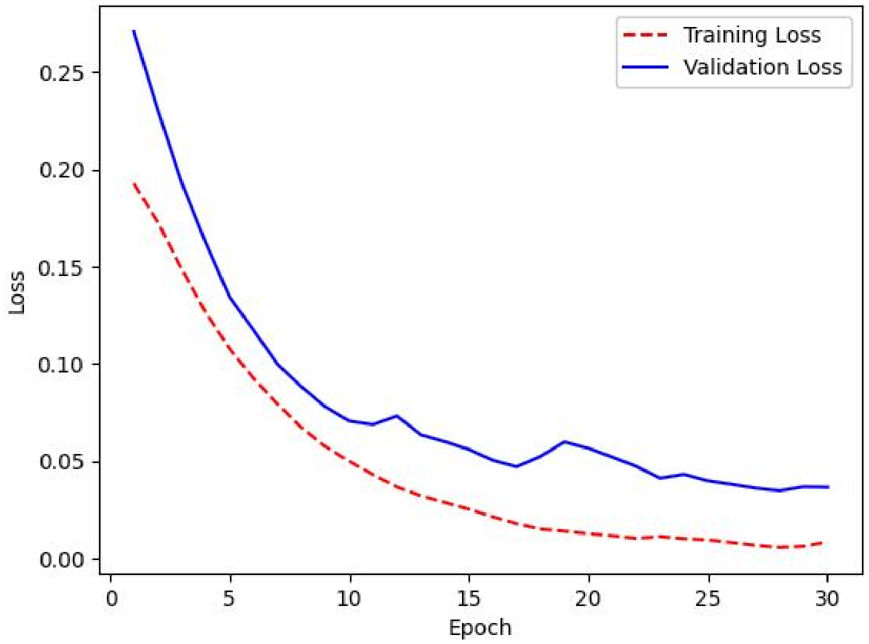


Figure 3.29: train and validation loss against epoch count

4.7.3.4 Ling Linguist

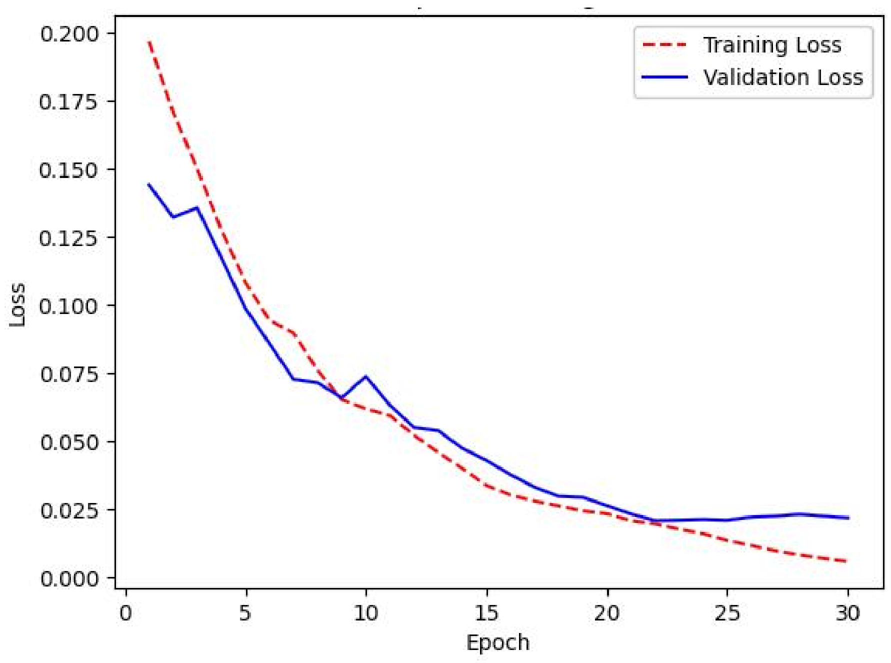


Figure 3.30: train and validation loss against epoch count

4.7.3.5 Interpretation of results:

By viewing the classification reports of our DistilBERT model and the train and validation loss against epoch, we can determine several key characteristics of the model's behavior.

		Enron	Spamassassin	SMS spam	Ling linguist
accuracy		98.8273%	99.1628%	99.2832%	99.6552%
F1-score		98.8273%	99.3502%	99.5859%	99.7955%
Precision	0	98.5461%	99.7101%	99.1753%	100.0%
	1	99.1003%	98.1818%	100%	97.8261%
recall	0	99.1100%	98.9928%	100%	99.5918%
	1	98.5304%	99.4737%	94.8052%	100.0%

Table 3.5: resulted metrics of the DistilBERT model

- **Model accuracy:**

The high accuracy across all datasets shows that the DistilBERT model operates effectively. But we need to further clarify by examining other metrics.

- **Class wise measures**
  - **f1-score**

Consistently high f1-scores across all datasets mean that the model is classifying well with little to no false positives or negatives.

- **Recall**

Overall high recall values on both classes with very small differences suggest that the model effectively captures the majority of instances for both classes without bias towards either class.

- **Precision**

High precision values for both classes with very little difference indicate that the model's positive predictions are mostly correct for both classes, implying a few false classifications.

Overfitting or underfitting indicators:

Closing the gap between training and validation loss:

<u>Significance:</u>

The model is proving to generalize well as we see the validation loss approaching the training loss.

<u>Implication:</u>

The model is learning useful patterns from the training data, and the validation loss decreasing is a sign of the model not overfitting.

Validation Loss Plateauing:

<u>Significance:</u>

If the validation loss stops decreasing, it means that the model learned best from the train data without fitting too close.

<u>Implication:</u>

The model learned its best from the training data, and further training will not improve performance in a noticeable manner but risk entering the overfitting stage.

Very Good Model Results:

Significance:

High accuracy, high precision, good recall, and F1-score, meaning that the model makes accurate predictions in both training and validation sets.

Implication:

The model's performance is strong; therefore, we can conclude that useful relations have been recognized by the model, and these can be applied to unseen data.

Conclusion:

The convergence of the training and validation losses, combined with very good outcomes from the models, indicate that a good balance has been struck between fitting to some particular elements within training data while generalizing for unseen items. Thus, this indicates a competent and well-trained mode.

4.7.4 Overall interpretation

DistilBERT surpasses BERT in every dataset, exhibiting a more remarkable F1-score and accuracy. It is because of this that our spam detection system should always choose DistilBERT over BERT, as it can tell spam emails from non-spam ones more accurately.
More observations we saw:
Our model yields exceptionally good results with all the datasets except Enron, trailing the metrics chart with 98.82% accuracy.

- Consistent high performance across varied datasets underscores our model's robustness and reliability, cementing its place as the model we will be using for further implementation (API and standalone web app).
- Our model outperformed all other related works in Enron, Spamassassin, SMS spam collection and ling-spam datasets.
- The most sensible hyperparameters were epoch count, dropout intensity, and L2 regulation strength, especially with the Enron dataset.

4.8 Deployment:

In deploying our spam detection system, we have structured our solution to be both robust and versatile, with the goal of meeting both current testing needs and future integration requirements. The deployment involves two main components: a standalone web application for model testing and retraining and an API backend designed to power mailbox spam detection.

4.8.1 Standalone Web Application:

We have developed a web application using Django to serve as a platform for testing potential spam emails and model-powering the site. This application offers a user-friendly interface where users can input their text to check if it is spam. The primary features of this web application include:

1. **Backend integration:**
   a. The web application is integrated with our state-of-the-art trained DistilBERT model.
   b. Upon receiving user input, the app processes the text using the saved model to predict whether the text is spam or not.
2. **Real-time response:**
   a. The prediction result is displayed to the user in real-time, offering quick and easy user experience.
3. **User Feedback Feature:**
   a. We have implemented a way for the user to provide feedback if they believe the model's prediction is incorrect.
   b. The user can indicate whether the text was spam or ham.
   c. The feedback is stored as an additional dataset to retrain the model, keeping it adapted to raising spam patterns.
4. **The feature of testing a whole CSV file, 'datasets':**
   a. The user may want to test many inputs at once; the test CSV file feature allows for time and performance savings.
5. **Export the database as CSV file:**
   a. This allows the developer to further advance the field of spam detection by not only keeping the model up-to-date with rising spam patterns but also submitting the dataset as learning and training material for researchers and students to work with.

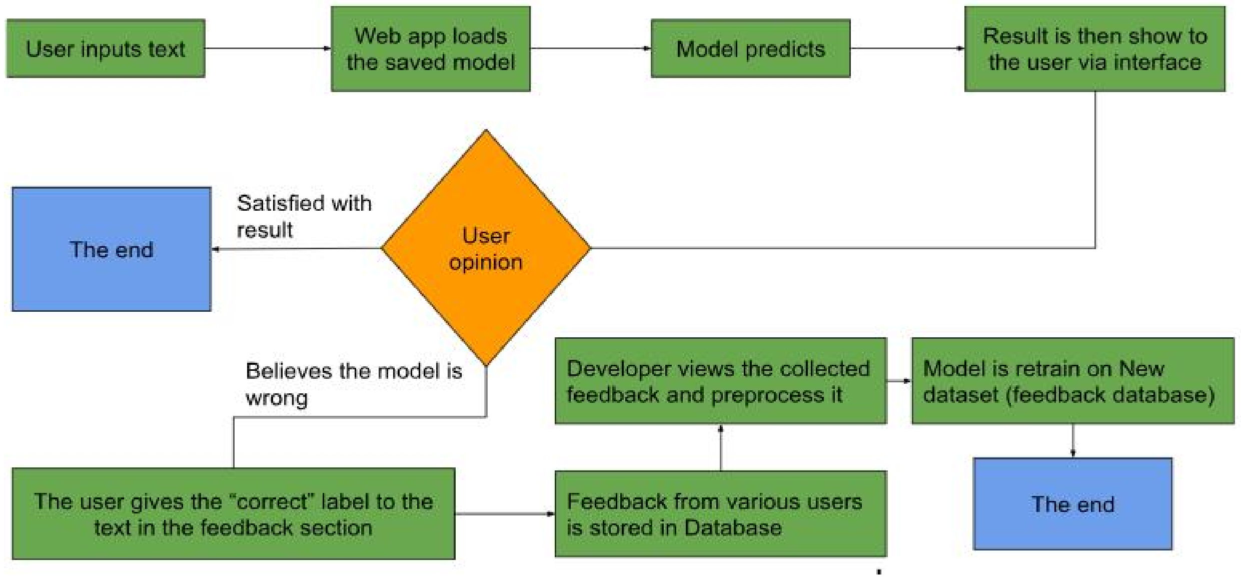

User inputs text
Web app loads the saved model
Model predicts
Result is then show to the user via interface
User opinion
Satisfied with result
The end
Believes the model is wrong
The user gives the "correct" label to the text in the feedback section
Feedback from various users is stored in Database
Developer views the collected feedback and preprocess it
Model is retrain on New dataset (feedback database)
The end

Figure 3.31: Web Application Workflow

4.8.1.1 Screenshots

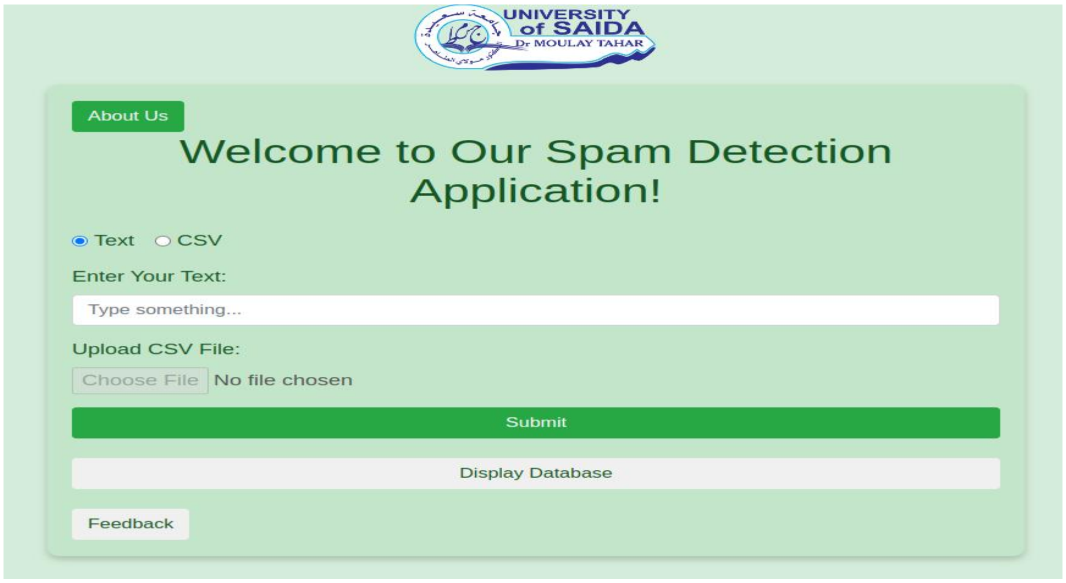
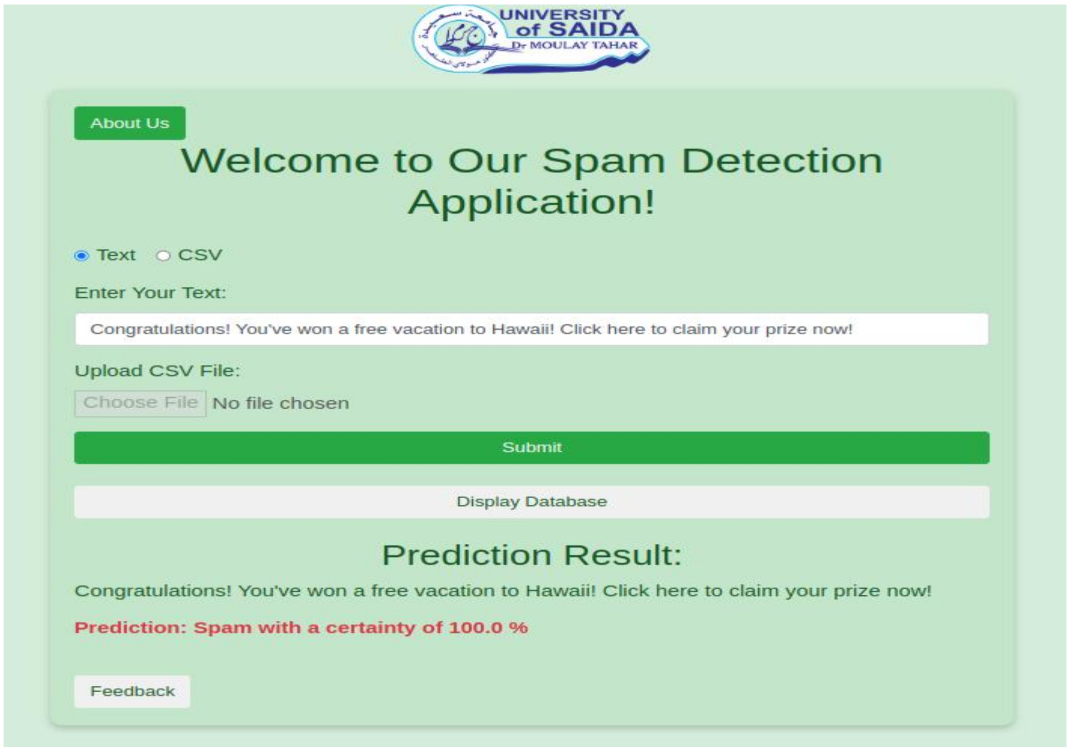


Figure 3.32: the web application interface



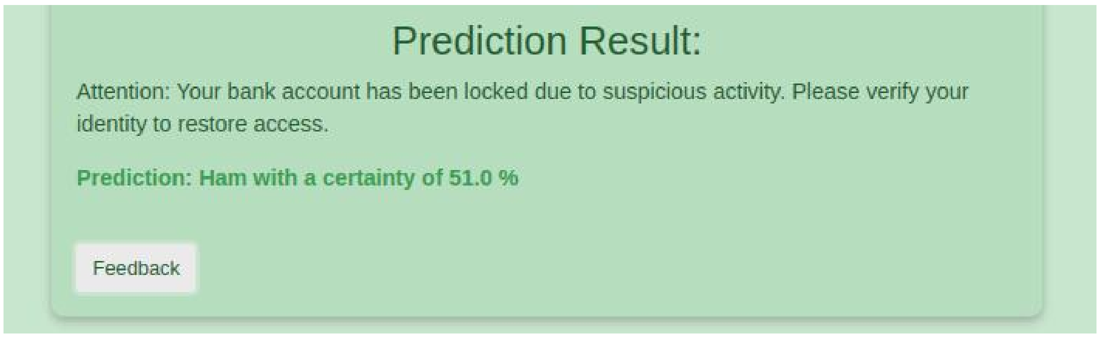
Congratulations! You've won a free vacation to Hawaii! Click here to claim your prize now!

Prediction Result:

Congratulations! You've won a free vacation to Hawaii! Click here to claim your prize now!

Prediction: Spam with a certainty of 100.0 %

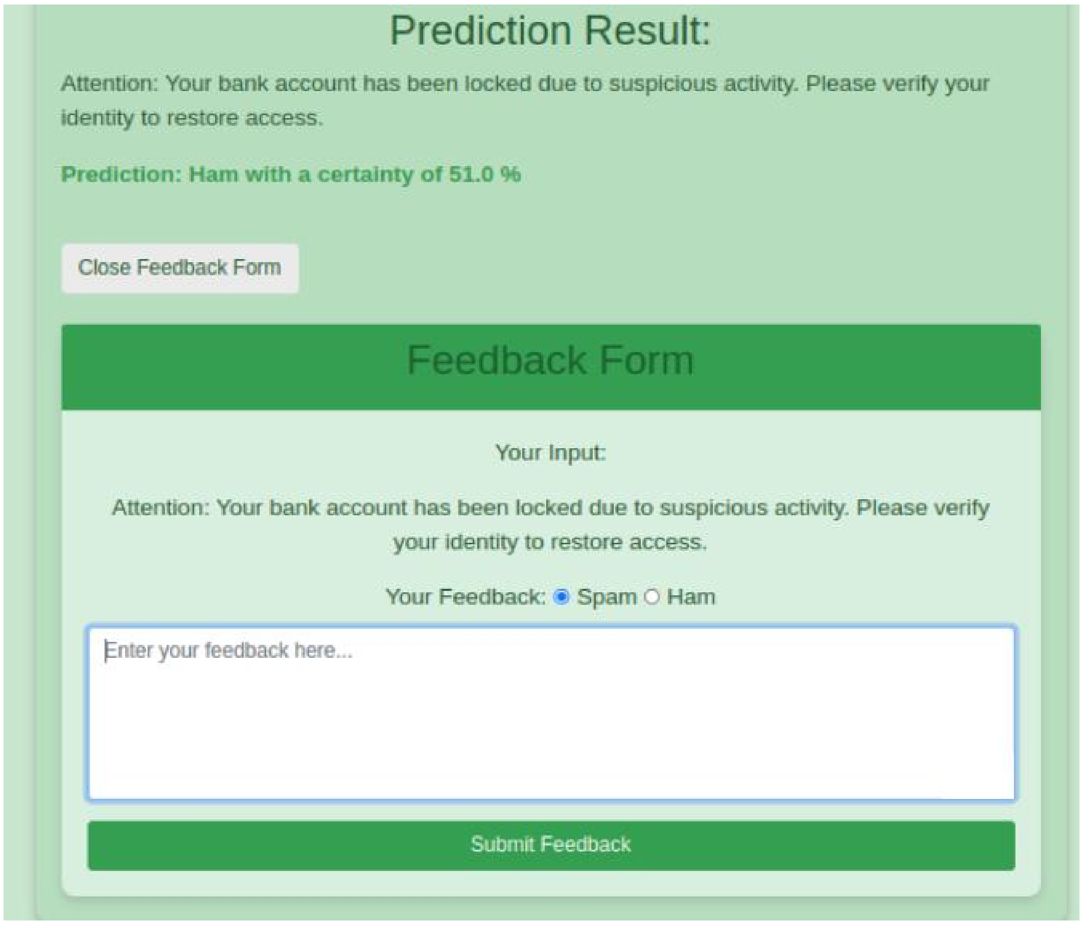
Figure 3.33: Example of Spam text


Prediction Result:

Attention: Your bank account has been locked due to suspicious activity. Please verify your identity to restore access.

Prediction: Ham with a certainty of 51.0 %

Figure 3.34: Example of Wrong Prediction


Prediction Result:

Attention: Your bank account has been locked due to suspicious activity. Please verify your identity to restore access.

Prediction: Ham with a certainty of 51.0 %

Your Input:

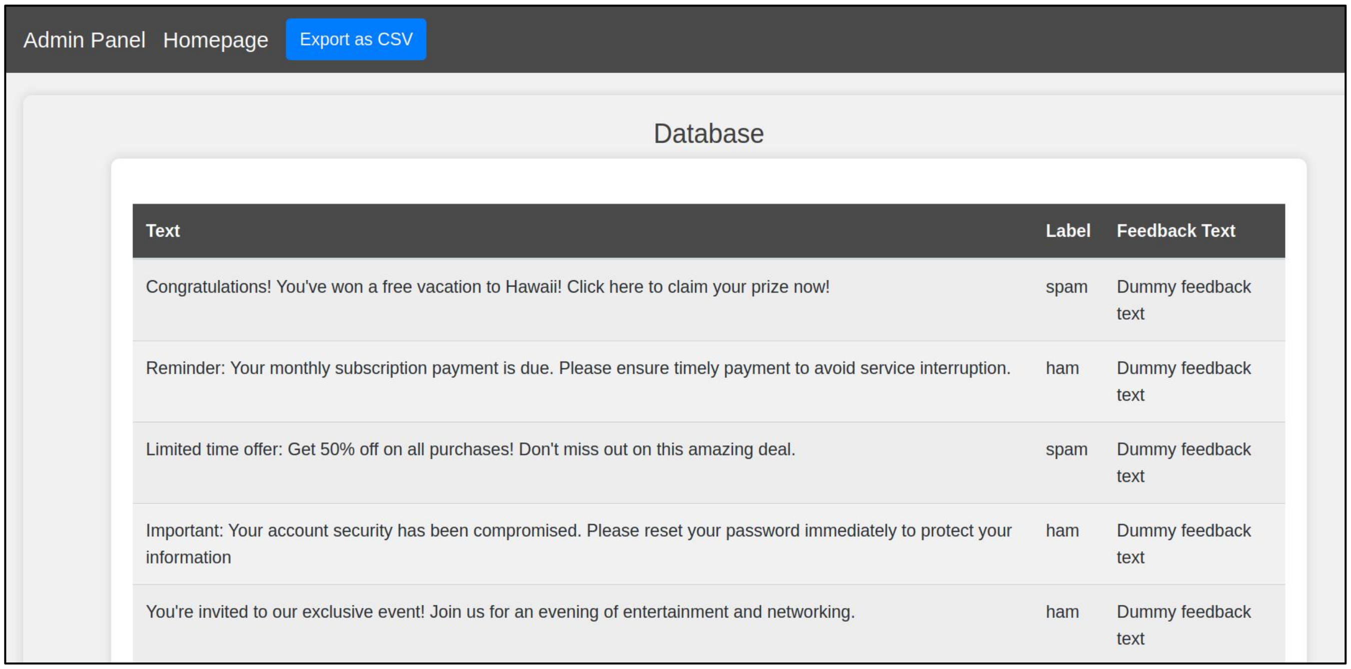
Attention: Your bank account has been locked due to suspicious activity. Please verify your identity to restore access.

Your Feedback: [✓ Spam] Ham

Figure 3.35: Feedback Form with the Correct Label

4.8.1.2 Components of a Feedback Form:

1. **The text that was entered by the user:**
   This displays the text in question for the user to review and provide feedback on.
2. **The correct label selector:**
   For the user to select the correct label for the text.
3. **Feedback description:**
   to explain their reasoning behind the selected label. This part is for the developer to see.

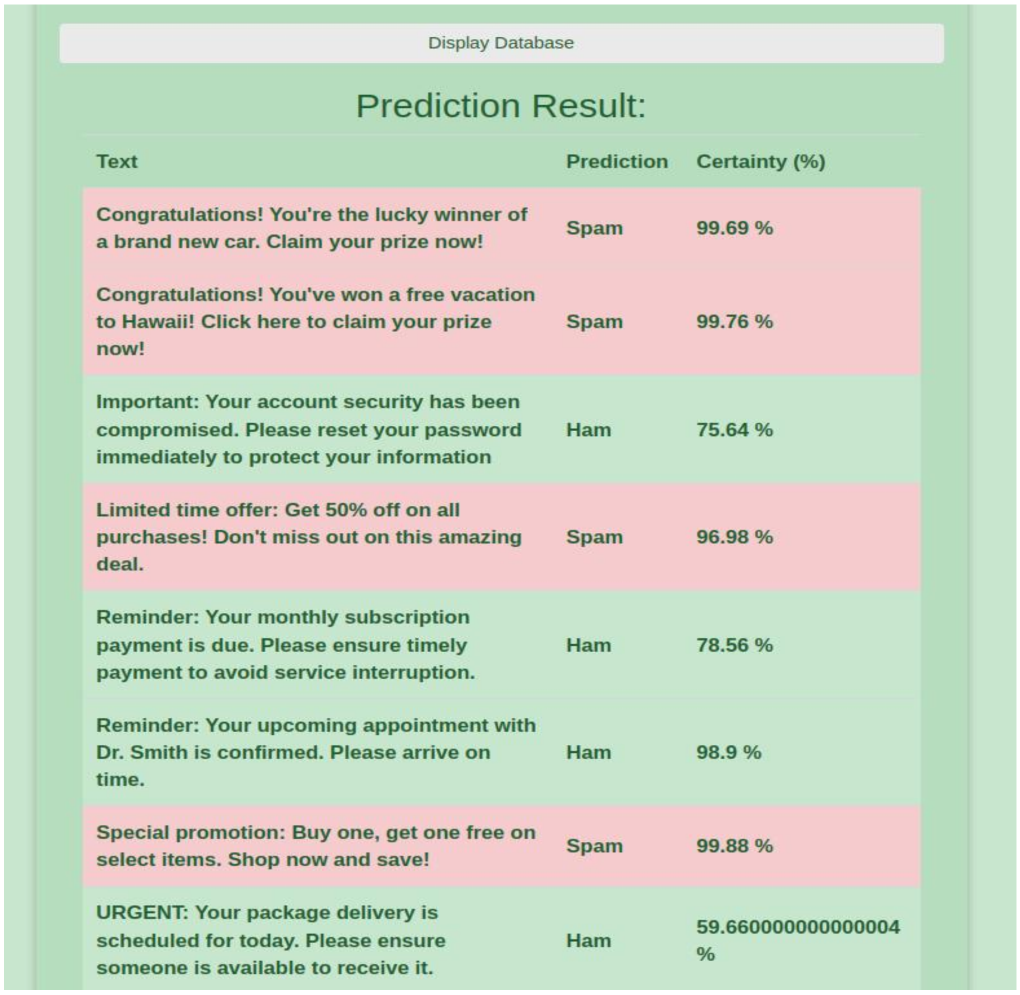


Database

Text	Label	Feedback Text
Congratulations! You've won a free vacation to Hawaii! Click here to claim your prize now!	spam	Dummy feedback text
Reminder: Your monthly subscription payment is due. Please ensure timely payment to avoid service interruption.	ham	Dummy feedback text
Limited time offer: Get 50% off on all purchases! Don't miss out on this amazing deal.	spam	Dummy feedback text
Important: Your account security has been compromised. Please reset your password immediately to protect your information	ham	Dummy feedback text
You're invited to our exclusive event! Join us for an evening of entertainment and networking.	ham	Dummy feedback text

Figure 3.36: Database interface

The content of the database can be exported as a CSV file. We use it to further retrain the model to keep up with rising patterns of spam and publish new datasets after thorough examination to spin the wheel of development in the spam detection field.



Prediction Result:

Text	Prediction	Certainty (%)
Congratulations! You're the lucky winner of a brand new car. Claim your prize now!	Spam	99.69 %
Congratulations! You've won a free vacation to Hawaii! Click here to claim your prize now!	Spam	99.76 %
Important: Your account security has been compromised. Please reset your password immediately to protect your information	Ham	75.64 %
Limited time offer: Get 50% off on all purchases! Don't miss out on this amazing deal.	Spam	96.98 %
Reminder: Your monthly subscription payment is due. Please ensure timely payment to avoid service interruption.	Ham	78.56 %
Reminder: Your upcoming appointment with Dr. Smith is confirmed. Please arrive on time.	Ham	98.9 %
Special promotion: Buy one, get one free on select items. Shop now and save!	Spam	99.88 %
URGENT: Your package delivery is scheduled for today. Please ensure someone is available to receive it.	Ham	59.660000000000004 %

Figure 3.37: Results of dataset prediction

4.8.2 API

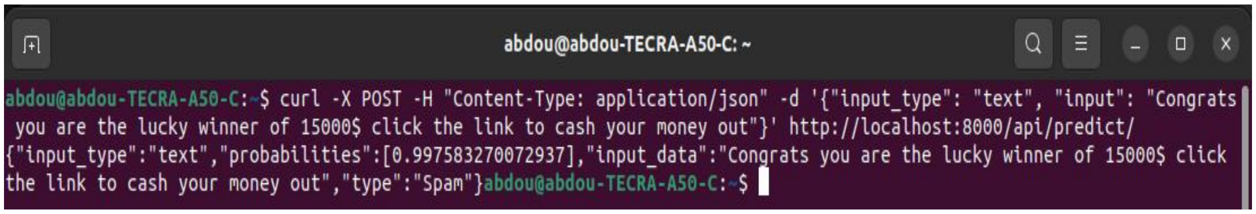
The Spam Detection API is designed to classify text inputs as either "Spam" or "Ham" using our trained DistilBERT model. This RESTful API offers a simple and efficient way for users to integrate spam detection capabilities into their applications. The API supports both individual text inputs and bulk predictions via CSV files, providing flexibility for various use cases.

```
{
 "input_type": "text",
 "input": "Your text here"
}
```

Figure 3.38: Request payload example

```
{
 "input_type": "text",
 "probabilities": [0.85],
 "input_data": "Your text here",
 "type": "Spam"
}
```

Figure 3.39: Expected response



abdou@abdou-TECRA-A50-C: ~

```
abdou@abdou-TECRA-A50-C:~$ curl -X POST -H "Content-Type: application/json" -d '{"input_type": "text", "input": "Congrats
 you are the lucky winner of 15000$ click the link to cash your money out"}' http://localhost:8000/api/predict/
{"input_type":"text","probabilities":[0.997583270072937],"input_data":"Congrats you are the lucky winner of 15000$ click
the link to cash your money out","type":"Spam"}abdou@abdou-TECRA-A50-C:~$
```

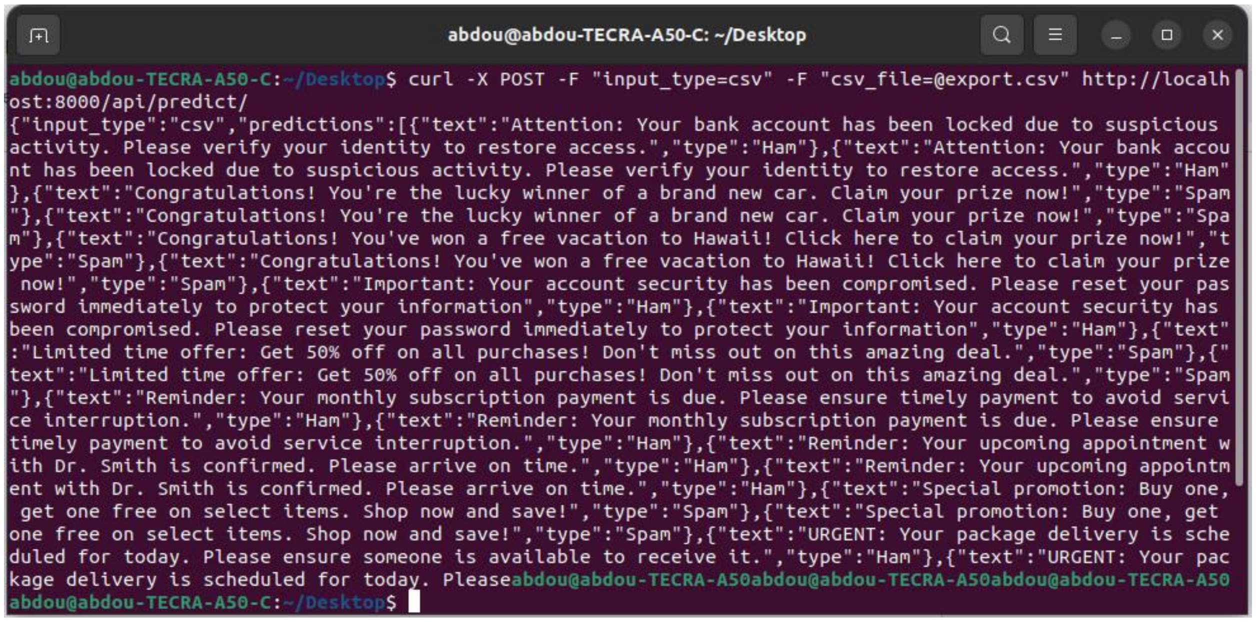
Figure 3.40: example using curl to test a text with response

```
curl    -X    POST    -F    "input_type=csv"    -F    "csv_file=@path_to_file.csv"
http://192.168.62.28:8000/api/predict/
```

Figure 3.41: Example of curl to test csv

```
{
 "input_type": "csv",
 "predictions": [
   {"text": "This is a sample email text 1", "type": "Ham"},
   {"text": "This is a sample email text 2", "type": "Ham"},
   {"text": "Free money!!! Click here now!", "type": "Spam"},
   {"text": "Meeting at 3 PM tomorrow", "type": "Ham"}
 ]
}
```

Figure 3.42: Expected response with CSV input



abdou@abdou-TECRA-A50-C: ~/Desktop

```
abdou@abdou-TECRA-A50-C:~/Desktop$ curl -X POST -F "input_type=csv" -F "csv_file=@export.csv" http://localh
ost:8000/api/predict/
{"input_type":"csv","predictions":[{"text":"Attention: Your bank account has been locked due to suspicious
activity. Please verify your identity to restore access.","type":"Ham"},{"text":"Attention: Your bank accou
nt has been locked due to suspicious activity. Please verify your identity to restore access.","type":"Ham"
},{"text":"Congratulations! You're the lucky winner of a brand new car. Claim your prize now!","type":"Spam
"},{"text":"Congratulations! You're the lucky winner of a brand new car. Claim your prize now!","type":"Spa
m"},{"text":"Congratulations! You've won a free vacation to Hawaii! Click here to claim your prize now!","t
ype":"Spam"},{"text":"Congratulations! You've won a free vacation to Hawaii! Click here to claim your prize
 now!","type":"Spam"},{"text":"Important: Your account security has been compromised. Please reset your pas
sword immediately to protect your information","type":"Ham"},{"text":"Important: Your account security has
been compromised. Please reset your password immediately to protect your information","type":"Ham"},{"text"
:"Limited time offer: Get 50% off on all purchases! Don't miss out on this amazing deal.","type":"Spam"},{"
text":"Limited time offer: Get 50% off on all purchases! Don't miss out on this amazing deal.","type":"Spam
"},{"text":"Reminder: Your monthly subscription payment is due. Please ensure timely payment to avoid servi
ce interruption.","type":"Ham"},{"text":"Reminder: Your monthly subscription payment is due. Please ensure
timely payment to avoid service interruption.","type":"Ham"},{"text":"Reminder: Your upcoming appointment w
ith Dr. Smith is confirmed. Please arrive on time.","type":"Ham"},{"text":"Reminder: Your upcoming appointm
ent with Dr. Smith is confirmed. Please arrive on time.","type":"Ham"},{"text":"Special promotion: Buy one,
 get one free on select items. Shop now and save!","type":"Spam"},{"text":"Special promotion: Buy one, get
one free on select items. Shop now and save!","type":"Spam"},{"text":"URGENT: Your package delivery is sche
duled for today. Please ensure someone is available to receive it.","type":"Ham"},{"text":"URGENT: Your pac
kage delivery is scheduled for today. Pleaseabdou@abdou-TECRA-A50abdou@abdou-TECRA-A50abdou@abdou-TECRA-A50
abdou@abdou-TECRA-A50-C:~/Desktop$
```

Figure 3.43: Example of CSV test along with result

4.8.2.1 Usage Examples for Integrating the Spam Detection API

1. **Email client integration:**
   Email clients can use the API to automatically filter incoming emails into spam and non-spam categories, providing users with a cleaner inbox experience.
2. **Mobile Applications:**
   Mobile app developers can integrate the API to scan text inputs from users, such as chat messages or comments, for spam detection, enhancing the overall user experience by minimizing unwanted content.

Conclusion

This work presents a multifaceted solution to the pervasive challenge of spam emails. Leveraging a highly accurate DistilBERT model, we developed a user-centric web app and a developer-oriented API. This synergy empowers both end-users and developers to effectively identify and combat spam, fostering a more secure and user-friendly online experience.

Chapter Five

Future work and conclusion

Chapter 5 Future work and Conclusion

5.1 Future work

This type of study can be expanded considerably with more time and resources. Availability of more complex and abundant data can enhance the model's applicability to different kinds of spam patterns. Training the bigger and complex models also might improve the detection accuracy even more. More time spent on fine-tuning and applying different hyperparameters would benefit the final model by increasing its efficiency. These, would contribute to a more powerful and reliable spam detection system.The future research can look into several promising directions. Additional preprocessing on the data obtained from the method we discussed for collecting feedbacks could further contribute its utility to studying and improving methods behind spam filtering. Improving more on the given API architectural structures would considerably improve the integration of the system. For instance, the application of continuous improvement processes would ensure effectiveness of the model against continuously emerging spam strategies. Further research can be viable by exploring hybrid detection models that incorporate BERT in conjunction with other algorithms or methods. Improving the possibilities of multilingual spam identification.

5.2 Conclusion

This thesis offers an extensive solution to the spam detection problem using BERT and its variant model to enhance the efficiency compared to conventional techniques. This work showed improved performance in comparison with several benchmarks related to different datasets, resulting from our fine-tuning of configurations and hyperparameters. The implementation of practical applications was through an easy to use web based application with feedback which would enable changes to be made as advances are being made and through a developer API for integration in other apps.

References

[1] : Kulikova, T. (2024, March 7). Spam and phishing in 2023. Kaspersky. Retrieved May 12, 2024, from https://securelist.com/spam-phishing-report-2023/112015/

[2] : Staveley, C. (n.d.). Council Post: The Power Of Emotions: How Cybercriminals Are Taking Advantage Of Psychology. Forbes. Retrieved May 20, 2024, from https://www.forbes.com/sites/forbestechcouncil/2022/10/10/the-power-of-emotions-how-cybercriminals-are-take-advantage-of-psychology/?sh=242b77443461

[3] : Moorthy, J. (2024, April 26). 23 Email spam statistics to know in 2024. Mailmodo. Retrieved May 1, 2024, from https://www.mailmodo.com/guides/email-spam-statistics/

[4] : November 2008, A. H. 10. (n.d.). TechRadar. retrieved in May 20 ,2024 https://www.techradar.com/news/internet/computing/spam-gets-1-response-per-12-500-000-emails-483381

[5] : Ma, X., Fang, G., & Wang, X. (2023). Llm-pruner: On the structural pruning of large language models. Advances in neural information processing systems, 36, 21702-21720.

[6] : Eng Emad Eldin Ibrahim, B. (2024, January 9). AI-ML-DML-GEN AI- LLM sorting according to prices. Linkedin.com. https://www.linkedin.com/pulse/ai-ml-dml-gen-ai-llm-sorting-according-prices-moselhy-ibrahim-bakr-7mz2f retrieved in 20/05/2024

[7] : Daniel. (2023, June 9). Natural language processing (NLP): Definition and principles. Data Science Courses | DataScientest; DataScientest. https://datascientest.com/en/natural-language-processing-definition-and-principles retrieved in 20/05/2024

[8] : Neural Network Architecture: all you need to know as an MLE [2023 edition]. (n.d.). Kili-website. Retrieved May 7, 2024, from https://kili-technology.com/data-labeling/machine-learning/neural-network-architecture-all-you-need-to-know-as-an-mle-2023-edition#-what-is-a-neuron?

[9] : Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., ... & Polosukhin, I. (2017). Attention is all you need. Advances in neural information processing systems, 30.

[10] : Liu, W., Zhou, P., Zhao, Z., Wang, Z., Ju, Q., Deng, H., & Wang, P. (2019). K-BERT: Enabling Language Representation with Knowledge Graph. arXiv e-prints. arXiv preprint arXiv:1909.07606.

[11] : Yang, Z., Dai, Z., Yang, Y., Carbonell, J., Salakhutdinov, R. R., & Le, Q. V. (2019). Xlnet: Generalized autoregressive pretraining for language understanding. Advances in neural information processing systems, 32.
[12] : Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2018). Bert: Pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805.
[13] : Sun, J., Liu, Y., Cui, J., & He, H. (2022). Deep learning-based methods for natural hazard named entity recognition. Scientific reports, 12(1), 4598.
[14] : Koroteev, M. V. (2021). BERT: a review of applications in natural language processing and understanding. arXiv preprint arXiv:2103.11943.
[15] : Trung, N. D., Ngoc, T. T., & Huynh, H. X. (2019). Automated pneumonia detection in x-ray images via depthwise separable convolution based learning. Proc FAIR-Fundament Appl IT Res. https://doi. org/10.15625/vap.
[16] : Lample, G., & Conneau, A. (2019). Cross-lingual language model pretraining. arXiv preprint arXiv:1901.07291.
[17] : Zhang, T., Wu, F., Katiyar, A., Weinberger, K. Q., & Artzi, Y. (2020). Revisiting few-sample BERT fine-tuning. arXiv preprint arXiv:2006.05987
[18] : Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter. arXiv preprint arXiv:1910.01108.
[19] : Zamil, Y. K., Ali, S. A., & Naser, M. A. (2019). Spam image email filtering using K-NN and SVM. International Journal of Electrical and Computer Engineering (IJECE), 9(1), 245-254.
[20] : Reddy, K. N., & Kakulapati, V. (2021). Classification of Spam Messages using Random Forest Algorithm. Journal of Xidian University, 15(8), 495-505.
[21] : Gupta, M., Bakliwal, A., Agarwal, S., & Mehndiratta, P. (2018, August). A comparative study of spam SMS detection using machine learning classifiers. In 2018 eleventh international conference on contemporary computing (IC3) (pp. 1-7). IEEE.
[22] : Rodrigues, A. P., Fernandes, R., Shetty, A., K, A., Lakshmanna, K., & Shafi, R. M. (2022). [Retracted] Real-Time Twitter Spam Detection and Sentiment Analysis using Machine Learning and Deep Learning Techniques. Computational Intelligence and Neuroscience, 2022(1), 5211949.
[23] : Cao, J., & Lai, C. (2020, December). A bilingual multi-type spam detection model based
on M-BERT. In GLOBECOM 2020-2020 IEEE Global Communications Conference (pp. 1-6).
IEEE.

[24] : Lee, Y., Saxe, J., & Harang, R. (2020). CATBERT: Context-aware tiny BERT for detecting social engineering emails. arXiv preprint arXiv:2010.03484.

[25] : Barsever, D., Singh, S., & Neftci, E. (2020, July). Building a better lie detector with BERT: The difference between truth and lies. In 2020 International Joint Conference on Neural Networks (IJCNN) (pp. 1-7). IEEE.

[26] : Sahmoud, T., & Mikki, D. M. (2022). Spam detection using BERT. arXiv preprint arXiv:2206.02443.

[27] : Oswald, C., Simon, S. E., & Bhattacharya, A. (2022). Spotspam: Intention analysis–driven sms spam detection using bert embeddings. ACM Transactions on the Web (TWEB), 16(3), 1-27

[28] : Tida, V. S., & Hsu, S. (2022). Universal spam detection using transfer learning of BERT model. arXiv preprint arXiv:2202.03480.

[29] : Godbole Siddharth, Grubinska, K., & Kelnreiter, O. (2020, February 7). Economic uncertainty identification. Retrieved May 2, 2024, from https://humboldtwi.github.io/blog/research/information_systems_1920/uncertainty_identification_transformers/

[30] : kaggle : BERT https://www.kaggle.com/models/tensorflow/bert

[31] : Evtimov, R., Falli, M., & Maiwald, A. (2020, February). Anti Social Online Behaviour Detection with BERT. Retrieved May 1, 2024, from https://humboldt-wi.github.io/blog/research/information_systems_1920/bert_blog_post

[32] : Srivastava, N., Hinton, G., Krizhevsky, A., Sutskever, I., & Salakhutdinov, R. (2014). Dropout: a simple way to prevent neural networks from overfitting. The journal of machine learning research, 15(1), 1929-1958.

[33] : Kaggle : distilbert https://www.kaggle.com/models/jeongukjae/distilbert

[34] : Adel, H., Dahou, A., Mabrouk, A., Abd Elaziz, M., Kayed, M., El-Henawy, I. M., ... & Amin Ali, A. (2022). Improving crisis events detection using distilbert with hunger games search algorithm. Mathematics, 10(3), 447.

[35] : Cortes, C., Mohri, M., & Rostamizadeh, A. (2012). L2 regularization for learning kernels. arXiv preprint arXiv:1205.2653.

[36] : Ioffe, S., & Szegedy, C. (2015, June). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In International conference on machine learning (pp. 448-456). Pmlr.

[37] : huggingface , SetFit's Enron spam collection

[38] : Kaggle , Mandy Gu's Ling-Spam Dataset

[39] : Kaggle , UCI's SMS Spam Collection Dataset

[40] : huggingface , Talby's Spamassassin dataset

[41] : Ogbemi, M. (2023, October 16). What is Overfitting in Machine Learning? freeCodeCamp.org. Retrieved May 3, 2024, from https://www.freecodecamp.org/news/what-is-overfitting-machine-learning/#:~:text=Our%20training%20dataset%20contains%2080%2C000,we%20have%20an%20overfitting%20problem.

[42] : GeeksforGeeks. (2024, March 11). ML Underfitting and overfitting. GeeksforGeeks. Retrieved May 7, 2024, from https://www.geeksforgeeks.org/underfitting-and-overfitting-in-machine-learning/

[43] : Jason Brownlee. (2020, January). How to diagnose overfitting and underfitting of LSTM models. Retrieved May 16, 2024, from https://machinelearningmastery.com/diagnose-overfitting-underfitting-lstm-models/