AUGMENTING A TURKISH DATASET FOR SPAM FILTERING USING NATURAL LANGUAGE PROCESSING TECHNIQUES

A THESIS SUBMITTED TO

THE GRADUATE SCHOOL OF INFORMATICS INSTITUTE OF

THE MIDDLE EAST TECHNICAL UNIVERSITY

BY

AYŞENUR AKSOY

IN PARTIAL FULFILLMENT OF THE REQUIREMENTS FOR THE

DEGREE OF

MASTER OF SCIENCE

IN

THE DEPARTMENT OF CYBERSECURITY

AUGUST 2022

Approval of the thesis:

AUGMENTING A TURKISH DATASET FOR SPAM FILTERING

USING NATURAL LANGUAGE PROCESSING TECHNIQUES

Submitted by Ayşenur AKSOY in partial fulfillment of the requirements for the degree of **Master of Science in Cyber Security Department, Middle East Technical University** by,

Prof. Dr. Deniz Zeyrek Bozşahin
Dean, **Graduate School of Informatics** _____

Assoc. Prof. Dr. Cihangir Tezcan
Head of Department, **Cyber Security** _____

Prof. Dr. Banu Günel Kılıç
Supervisor, **Information Systems Dept., METU** _____

Assoc. Prof. Dr. Cengiz Acartürk
Co-Supervisor, **Cognitive Science Dept., METU** _____

Examining Committee Members:

Assoc. Prof. Dr. Cihangir Tezcan
Cyber Security Dept., METU _____

Prof. Dr. Banu Günel Kılıç
Information Systems Dept., METU _____

Assoc. Prof. Dr. Burcu Can
Research Institute in Information and
Language Processing., University of Wolverhampton _____

Date: **25.08.2022**

I hereby declare that all information in this document has been obtained and presented in accordance with academic rules and ethical conduct. I also declare that, as required by these rules and conduct, I have fully cited and referenced all material and results that are not original to this work.

Name, Last name : A y ş e n u r A K S O Y

Signature : _____

ABSTRACT

AUGMENTING A TURKISH DATASET FOR SPAM FILTERING

USING NATURAL LANGUAGE PROCESSING TECHNIQUES

Aksoy, Ayşenur

MSc., Department of Cyber Security

Supervisor: Prof. Dr. Banu Günel Kılıç

Co-Supervisor: Assoc. Prof. Dr. Cengiz ACARTÜRK

August 2022, 85 pages

Today, how we communicate is altering as a consequence of the evolution of the internet. Since one of the main communication ways of the internet is e-mail systems and they are easy to use, cheap and fast, and have a wide user base, they have also become a broad environment for malicious actors to act within. Correspondingly, spam e-mails, defined as any kind of unwanted, unwelcomed e-mails sent in bulk, are one of the main tools for these malicious actors. Even if there is not yet a definitive way to stop spam e-mails, filtering techniques are improving all the time. In time, spam filtering became one of the most commonly used text classification issues in Natural Language Processing, too. There are multiple ways to improve the classification success of the machine learning methods, one of them is data augmentation. Augmentation serves to generate more unique data from the dataset at hand and improves the functionality and accuracy of machine learning models. A machine learning model improves if the dataset is sufficient and large enough. In this study, we examined the effects of semantically augmenting a Turkish dataset on the accuracy of spam filtering methods and observed efficient results that can be used in research.

Keywords: Spam filtering, NLP on Turkish, Data Augmentation

ÖZ

DOĞAL DİL İŞLEME TEKNİKLERİ KULLANILARAK SPAM

FİLTRELEME İÇİN TÜRKÇE VERİ KÜMESİNİN GENİŞLETİLMESİ

Aksoy, Ayşenur

Yüksek Lisans, Siber Güvenlik Bölümü
Tez Yöneticisi: Prof. Dr. Banu Günel Kılıç

Eş Tez Yöneticisi: Doç. Dr. Cengiz ACARTÜRK

Ağustos 2022, 85 sayfa

Günümüzde, internetin evriminin bir sonucu olarak iletişim kurma şeklimiz de değişiyor. İnternetin temel iletişim yollarından biri olan e-posta sistemleri; kullanımının kolay, ucuz ve hızlı olması ve geniş bir kullanıcı kitlesine sahip olması, kötü niyetli aktörlerin de içinde hareket edebileceği geniş bir ortam haline gelmiştir. Buna bağlı olarak istenmeyen ve toplu olarak gönderilen her türlü e-posta olarak tanımlanan spam e-postalar, internetteki kötü niyetli aktörlerin başlıca araçlarından biri haline gelmiştir. İstenmeyen e-postaları durdurmanın henüz kesin bir yolu olmasa da, filtreleme teknikleri her zaman gelişmeye devam etmektedir. Dolayısıyla, istenmeyen e-posta filtreleme, Doğal Dil İşleme'de de en sık kullanılan metin sınıflandırma konularından biri haline geldi. Bu amaçla kullanılan makine öğrenme yöntemlerinin sınıflandırma başarısını artırmanın ise birden çok yolu vardır ve veri artırma bunlardan biridir. Artırma, eldeki veri kümesinden daha fazla veri ve örnek oluşturmaya hizmet eder ve eğitim veri kümelerine benzersiz örnekler ekleyerek makine öğrenme modellerinin işlevselliğini ve doğruluğunu artırır. Veri kümesi yeterli ve yeterince büyükse, makine öğrenme modeli de daha iyi performans gösterir. Bu çalışmada, Türkçe bir veri setini anlamsal olarak büyütmenin spam filtreleme yöntemlerinin doğruluğuna etkisini inceledik ve araştırmalarda kullanılabilecek verimli sonuçlar gözlemledik.

Anahtar Sözcükler: Spam filtreleme, Türkçe’de NLP, Veri Artırımı

To My Family…

ACKNOWLEDGEMENTS

First of all, I would like to thank my supervisors, Prof. Dr. Banu Günel KILIÇ for helping me get through this research and Assoc. Prof. Dr. Cengiz ACARTÜRK for guiding me throughout my entire study, continuously giving his support for me to get to this point.

I would also like to express my deepest gratitude to my committee members for their reviews and very helpful contributions; Assoc. Prof. Dr. Cihangir TEZCAN and Assoc. Prof. Dr. Burcu CAN, they let me benefit from their wide knowledge.

Last but not least, I would like to thank my family; my mother, my father and my siblings for always giving me moral support. Whenever things start to get tough, they were the ones that kept me going on my feet.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

LIST OF ABBREVIATIONS

CBOW	Continuous Bag of Words
BERT	Bidirectional Encoder Representations from Transformers
BOW	Bag of Words
DDoS	Distributed Denial of Service
DistilBERT	Distilled Bidirectional Encoder Representations from Transformers
GLUE	The General Language Understanding Evaluation
GPT	Generative Pre-Trained Transformer
HTML	Hypertext Markup Language
KNN	K Nearest Neighbour
NIST	National Institute of Standards and Technology
NLP	Natural Language Processing
SVM	Support Vector Machine
RNN	Recurrent Neural Network
TF-IDF	Term Frequency – Inverse Document Frequency
URL	Uniform Resource Locator
XGBOOST	eXtreme Gradient Boosting

CHAPTER 1

INTRODUCTION

The growth of the internet is changing the way we communicate and interact. It also gives malicious actors a broad environment to act within. Spam, which is one of the main tools for malicious actors on the internet, is defined as "The abuse of electronic messaging systems to indiscriminately send unsolicited bulk messages." by NIST (NIST Joint Task Force, 2020). As of today, most malicious domains, about 60%, are associated with spam campaigns (CISCO Secure, 2022). Adversaries can conduct non-targeted phishing, such as mass malware spam campaigns (ATT&CK Matrix for Enterprise, 2020). Phishing frequently takes the form of a spam e-mail combined with a malicious replica of an official website. In order to prevent internet users from getting spam e-mails, internet and e-mail providers started to use filters to distinguish between regular e-mails and spam e-mails according to a set of rules. Among all the techniques developed for detecting and preventing spam, filtering the e-mails is one of the most essential and prominent approaches (Ahmed et al., 2022). However, the rise in the volume of spam e-mails has created an intense need for the development of more dependable and robust anti-spam filters. Machine learning methods have been used to improve the spam filters and they are considered to be successful on it (Dada et al., 2019). Today, learning-based classifiers are commonly used for spam filtering.

The idea that computers can understand ordinary languages and hold conversations with human beings was predicted in a classic paper by Alan Turing in 1950 (Turing, 1950) as a hallmark of computational intelligence. Natural Language Processing (NLP), a subset of machine learning, enables computer systems to analyze and interpret texts. NLP provides communication between human language and computers. There are multiple uses and purposes of NLP and text classification is one of them. Spam filtering is one of the key applications of text classification.

To be able to use text classification for spam filtering, a large e-mail dataset is needed. A classifier model must be trained to label the dataset accordingly. Since English is known and accepted as the leading language on the internet, text-oriented studies are conducted mostly on English text. There are numerous studies on the English language and text in English and most of the NLP problems are considered solved and closed for English. However, in non-English languages, problems need unique approaches according to the morphology of the target language and Turkish also needs different approaches.

Recently, there has been a gap in the availability of publicly accessible e-mail datasets in Turkish for the usage of researchers to develop spam filtering methods systematically. To solve this problem, we created an e-mail dataset for the purpose of spam filtering in Turkish with Natural Language Processing techniques. We synthetically augmented the dataset according to word similarities and context. We prepared this dataset to serve researchers for them to conduct their studies in the field.

1.1. Motivation

Spam e-mail, also known as Unsolicited Commercial E-mail, is unsolicited and questionable mass-e-mailed content.

Between October 2020 and September 2021, global daily spam volume reached its highest point in July 2021, with almost 283 billion spam e-mails from a total of 336.41 billion sent e-mails. As of August 2021, this number dropped to 65.50 billion (CISCO Secure, 2022).

In the graphics in Figure 1, we can observe spam e-mails are always close in numbers to the regular e-mail traffic. It is also seen that the growth of e-mail traffic and spam e-mail traffic is directly proportional.

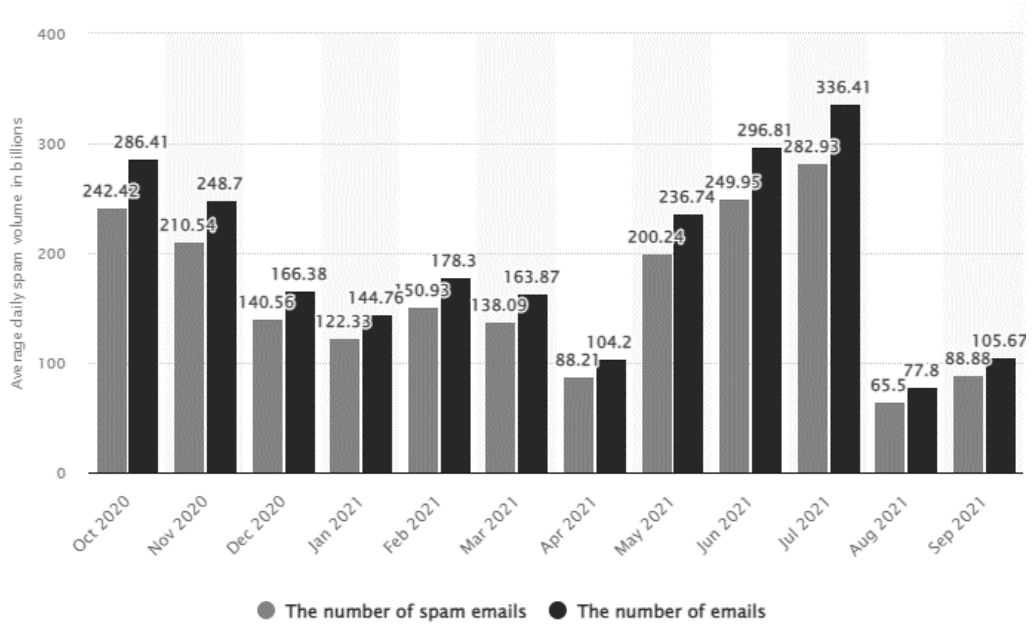


Figure 1: Average daily spam volume worldwide from October 2020 to September 2021 (in billions)

The most common method of spam is sent via e-mail, while it can also be distributed via text messages, social media, or phone calls. Adversaries can conduct non-targeted phishing, such as in mass malware spam campaigns.

Spam is inconvenient for internet users, consuming time and resources but spam is also a cybersecurity threat. According to annual report from Kaspersky lab, in 2021, cybercriminals involved in the creation and distribution of spam and phishing tried to lure users using a variety of topics such as lucrative investments, online streaming of global movie and TV premieres and themes related to restrictions, requirements (Kaspersky, 2022). While people think they can recognize spam even if it is not filtered as spam, spammers, those that send spam messages, regularly update their methods and messages to trick potential victims. In most cases, cyber-attackers choose the people as their target, not the systems, since people are easier to overcome. It can be said the most common method to target people is by sending spam e-mails and attaching phishing links to them based on the fact that phishing attacks account for more than 80% of reported security incidents. Phishing and similar fraud were the most prevalent cybercrime reported to the U.S. Internet Crime Complaint Center in 2021 (Federal Bureau of Investigation, 2021). Therefore, people are all constantly under the possibility of an attack from cybercriminals.

Because of these reasons, spam filtering has become an important matter and there have been a lot of attempts to solve it. In machine learning, spam filtering has also become one of the most popular text classification problem. In recent years, since the traditional rule-based filtering became lacking; word vector-based spam detection/filtering became a hot issue. Word vectors are basically the numeric equivalents of the words which are used for processing words. The studies done on this subject are widely focused on English since English is the "de-facto" language for technology and the internet. Google and Stanford University stated in joint research that e-mails attack are mostly in English. But the same research revealed a rise in phishing attacks which are not in English since the victim's language has an important role. Research stated that 78% of the e-mails targeting Japanese users getting the e-mails in Japanese language and 66% of the attacks that targeted users in Brazil were written in Portuguese (Simoiu et al., 2020). This leads the process of spam filtering with machine learning to be language-based too. In text-based academic research about this topic, Turkish databases for spam are usually small sized, collected personally and/or case-specific. This is why we looked for ways to get a larger dataset in Turkish synthetically and decided to augment the text data with NLP techniques to achieve that.

What is Augmentation and Why Augment the Data?

Data augmentation means synthesising new data from the data we already have. It means applying transformations to the original labelled data to create new data for the training. Hence, if we have data (X, Y), X is a sentence and Y is its corresponding label. In our context, X is the e-mail and Y is the spam or ham (non-spam) e-mail label.

As a part of data augmentation, X is transformed and X’ is created out of it, and the label is preserved.

(X, Y) ——T——> (X’, Y)

Since Y is still preserved with this transformation the e-mails X and X’ have to be semantically similar which means it should not change the meaning of the original sentence. Thus, even though X’ could be syntactically different compared to X, they should semantically mean the same thing. (Nithilaa Umasankar, 2021).

This is a concept that is used for a variety of data, such as sounds, images, or text. But since texts have to have complete meanings, unlike images which can have a meaning even if we cut them into half, text augmentation is considered one of the most difficult augmentation areas. Usually, the augmented data is similar to the existing data. Since the dataset is crucial for all machine learning problems, small or imbalanced datasets can become the actual problem of the process. And other than collecting more data, the solution for this seems to be augmentation.

Augmentation serves the purpose of generating more data and more examples from the dataset at hand without the effort to collect more relevant and usable real data. By creating additional and distinct instances for training datasets, data augmentation helps machine learning models perform better and produce more accurate results. A machine learning model improves if the dataset is large and sufficient. Reduced overfitting is one of the main benefits of data augmentation. For instance, a classification model trained on just three paragraphs will only be able to identify and categorize those specific texts and the data's generalizability will be improved by a few adjustments.

1.2. Research Question

In the literature, there are only a few e-mail datasets in Turkish to be used in spam filtering machine learning models. Since the cyber attackers started to specifically adjust to receivers’ language, it became more important to have more improved tools to handle spam traffic in Turkish as well.

In this study, our research questions are as follows:

- Can Turkish e-mail data be augmented semantically, with text representation methods in NLP? Text representation is converting words into numbers for machines to understand and decode patterns within a language. We aim to augment an e-mail dataset and create a larger dataset that is also meaningful in Turkish to use for spam filtering and text representations are needed for that. There are multiple ways of text representation in NLP. It can be done in different ways, such as frequency-based models like one-hot-encoding, count vector, co-occurrence, or TF-IDF. By using one-hot

encoding, categorical variables can be transformed into a format which machine learning algorithms can use to make more accurate predictions. In one hot encoding, words can be easily digitized; but when two different words are given, it is not possible to discover the relationships between these two words. It basically works in one way and requires a lot of memory. Next in order, count vectors are vectors created according to the frequency of words in the document and co-occurrence is a matrix which holds the number of words appearing together. It also requires large memory space and a lot of processing power. TF-IDF takes into account not only the rate of occurrence of words in the document but also the frequency of occurrence of a word in other documents. If it is a word or phrase which appears frequently in all documents, the TF-IDF value will be low. There are also content-based models such as word2vec. With word2vec, the key premise is that a word's meaning may be deduced from the company it keeps. Word2vec comes with two neural network architectures; CBOW (continuous bag of words) and Skip-gram. The destination word is predicted by CBOW using the nearby words as input. On the other hand, Skip-gram does the opposite job and predicts the words close to the input word of interest. There are also Transformers models (i.e. BERT, distilBERT, GPT), which are context-based. BERT, distilBERT or GPT models can produce many word embeddings for a word which properly represent its context or its position in a text. We aim to use models that capture the content with its meaning and the context.

- Is augmenting a dataset have an impact on the accuracy of spam filtering? The accuracy of text classification can be scored with different types of algorithms to measure the correctness of the machine learning model. Can we get a higher accuracy score with our augmented dataset for the text classification task of spam filtering? We hypothesize that the augmented dataset will provide improved accuracy to classify e-mails for spam filtering.

1.3. Organization of the Thesis

In this thesis, there are mainly four chapters that serve the purpose of forming an understanding of the conducted study.

- Chapter 1 gives a short introduction to the study, and the motivation and gives out the research question,

- Chapter 2 summarizes the background information needed for the process, gives out details about spam and spam as a cyber-threat, and the protection methods from it; the artificial intelligence/machine learning and natural language processing impacts on the spam issue in the perspective of our study,

- Chapter 3 presents the overall study process from collecting data to getting the results; it shows the steps of using different techniques like word2vec, BERT, GPT-2 to augment data semantically,

- Chapter 4 discusses the results of the methods chosen to augment the dataset and compares the results with accuracy values,

- Chapter 5 discusses the results of different augmentation techniques we used and their effects on the classification, specifies the limitations we came across and gives out some ideas as the future work of this study.

CHAPTER 2

BACKGROUND INFORMATION AND LITERATURE REVIEW

In this chapter, the terms, the main subjects, the methods, and the models used for our study will be described. First, we will start by describing e-mail, spam and why is spam a cybersecurity issue. Then we will explore spam filtering history and the machine learning application of spam filtering. Then we will focus on word embedding theories, and models for finding similar words according to cosine similarity and according to context. Then we will introduce the classification task in NLP.

For the English texts, there are annotated datasets and different packages such as NLTK (Natural Language Toolkit), TextBlob, or spaCy available online. Part of text-based research of the English text is considered a closed issue. Even though the natural language processing in English has its problems too, it can be handled quickly as there is a large community handling the same problems and sharing results and experiences. On the other hand, when the language to be processed has more flexible word order and richer morphology, it needs different ways of pre-processing, some extra functions to implement, and more time to achieve the same significant accuracy as the English language. We will be explaining language-specific practices for the Turkish language and wrap the chapter.

2.1. Understanding Spam and Why It Is a Cybersecurity Issue

Electronic mail (e-mail) is a digital letter sent over the internet, which is an easy and cheap way to communicate. Globally, as of 2019, a staggering 293.6 billion e-mails were sent each day and there are currently over 4 billion e-mail users worldwide (CISCO Secure, 2022).

An e-mail statistics report data shows the fact that the daily e-mail sent count has crossed 300 billion in 2020, with 319.6 billion in 2021, and a whopping 333.2 billion e-mails sent per day in 2022.

This means more than 3.5 million e-mails are sent per second, and the number is still growing as can be seen in Figure 2 (The Radicati Group, 2019).

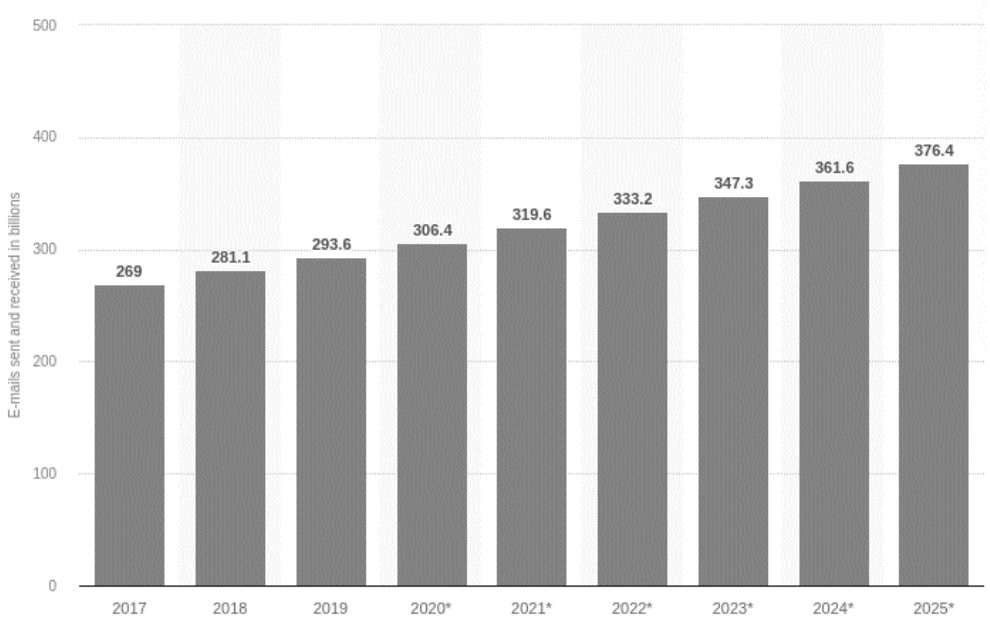


Figure 2: E-mails sent and received in billions from 2017 to 2025

Based on the research, it is estimated people will send and receive more than 376 billion e-mails per day by 2025 according to the data collected so far.

The same report shows e-mail traffic is growing over the years from 2015 to 2019 in Figure 3.

Daily Email Traffic	2015	2016	2017	2018	2019
Total Worldwide Emails Sent/Received Per Day (B)	**205.6**	**215.3**	**225.3**	**235.6**	**246.5**
% Growth		*5%*	*5%*	*5%*	*5%*
Business Emails Sent/Received Per Day (B)	**112.5**	**116.4**	**120.4**	**124.5**	**128.8**
% Growth		*3%*	*3%*	*3%*	*3%*
Consumer Emails Sent/Received Per Day (B)	**93.1**	**98.9**	**104.9**	**111.1**	**117.7**
% Growth		*6%*	*6%*	*6%*	*6%*

Figure 3: Daily e-mail traffic from 2015- to 2019

E-mail accounts can be opened from various sites which are providing this service. E-mail is an efficient way of communication as it saves a lot of time and money. These aspects make it a common communication tool in professional and personal communication. Programs called MAILBOX on the Massachusetts Institute of Technology (MIT) computers back in 1965 are the first examples of e-mail. Even though there were some forms of e-mail, a

networked system was created by ARPANET and e-mail was invented in 1972. The @ symbol is used because it was not in the names of the people. E-mails are defined as "username@computername". 75% of ARPANET traffic was sent by e-mail within a few years. With the invention of the e-mail, the world has made its way to the internet from ARPANET (Leiner et al., 2009).

The broadband internet subscribers in Türkiye, which were around 6 million in 2008, reached 88.8 million as of the first quarter of 2022. The annual rate of increase in the total number of internet subscribers was 5.9% (BTK - Information and Communication Technologies Authority, 2022).

Because it has a wide user base, is easy, cheap and fast, many companies popularly prefer e-mail systems to advertise, which results in unwanted advertisement or an active or passive attack type; spam.

As of today, spam is a term well-known by internet users. Any kind of unwanted, unwelcomed messages which are sent digitally and in bulk can be counted as spam. It is defined as "The abuse of electronic messaging systems to indiscriminately send unsolicited bulk messages." by NIST (NIST Joint Task Force, 2020). The first reported spam e-mail was sent by Gary Thuerk on May 3, 1978, to several hundred users on ARPANET. It was an advertisement for a presentation by Digital Equipment Corporation for their DECSYSTEM-20 products. In Figure 4 below, the history of digital spam can be seen (Ferrara, 2019).

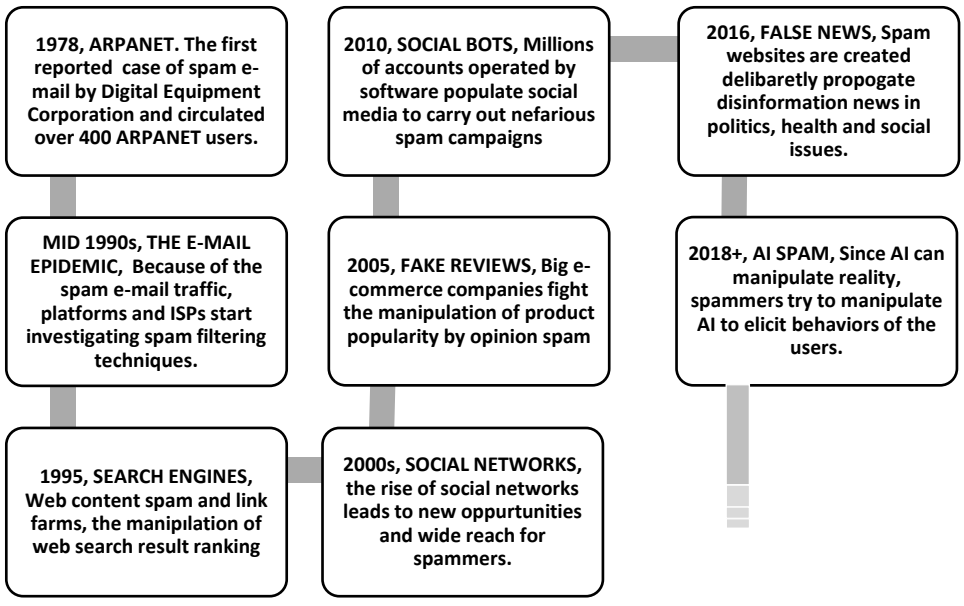


Figure 4: Timeline of the major milestones in the history of spam, from its inception to modern days

Spammers use several kinds of ways to bulk-send their unwanted messages. Some spam e-mails might have marketing aims and some other types of spam messages can spread malware, trick people into divulging personal information or scare them into thinking they need to pay to get out of trouble.

Spam causes a lot of trouble to the internet community, large amounts of spam traffic between servers cause delays in the delivery of legitimate e-mail, sorting out the unwanted messages takes time and introduces a risk of deleting normal e-mail by mistake, and people with dial-up internet access have to spend bandwidth downloading spam e-mail. There is quite an amount of pornographic spam that should not be exposed to children.

And last but not the least, spam can be an easy and powerful cyber-attack tool if it contains malicious links or asks for personal information. In a sample of more than 13 million e-mails identified as spam, more than 100,000 contained malicious attachments; nearly 1.4 million contained malicious web links. If opened, these attachments and links could infect the recipients' devices with software that allows cybercriminals to remotely access them (Alazab & Broadhurst, 2017). Spam protection is considered one of the main protections against cyber-attacks according to NIST since it can be the starting point for multiple types of cyber-attacks (NIST, 2020). A spam e-mail can turn the receiver's computer into a bot/zombie computer which can be used without their knowledge. Attackers can use the computer to create another mass spam e-mail campaign.

In order to fight spam; multiple ways are proposed; legal measures like the anti-spam law introduced in the US (The CAN-SPAM Act: Requirements for Commercial Emailers, 2004) or social methods such as educating e-mail users about spam and also other methods like blocking the known IP addresses of spammers. At last, there is spam filtering and various methods of it. An absolute solution to spam e-mails is yet to be found, and automatic filtering methods are being evolved day by day to fight against spam.

Spam filters abundantly block lots of spam e-mails but it remains a weight to the networks, e-mail servers, and overall internet. Given the astounding volume of spam that reaches e-mail inboxes, it is reasonable to believe there are global, structured, and virtual social networks of spammers. They target not only user e-mails but also those of entire nations and organizations. One of the tools in the informational war is spam (Nazirova, 2011). Even though the phrases "spam" and "war" have been used in the same context since 2003, (Gburzynski & Maitan, 2004; Weinstein, 2003) it is not until 2009 when the issue of spammers' social networks is discussed in academic studies. The method of spectral clustering is used in studies by Xu K.S., et al. to define and follow the social networks of spammers by tracking a set of spam communications gathered under project Honey Pot (Xu et al., 2009). They depict a spammer's social network as a graph with spammers as nodes, and they depict social connections between spammers as a corner between two graph junctions.

Spam filtering system research and development are being done intensively worldwide. Numerous businesses and organisations, in addition to academic institutions, are looking into and providing various theoretical, practical, and legal methods for spam filtering. Several organisations, including academic labs (i.e. CSAIL MIT in the United States, Computer Laboratory Faculty at

Cambridge University in the United Kingdom), research centres (i.e. IBM Research Center), and private businesses (Microsoft, Symantec, Kaspersky's Laboratory) had a hand in this process. Many international organizations pay close attention to the issue at hand. In 2003, the IETF (Internet Engineering Task Force) established the ASRG (AntiSpam Research Group). There have been numerous worldwide symposiums, summits, and conferences on this subject (Nazirova, 2011).

2.2. Artificial Intelligence and Machine Learning for Cyber Security

For general understanding, we will briefly introduce some key factors of artificial intelligence (AI) and machine learning, their applications in cyber security, and the critical parts for our study.

AI can be considered as the big picture. Its goal is to make it possible for computers to think as humans do, simulate human behaviour, and solve problems more quickly and effectively than people can. AI is capable of doing a wide range of functions, including planning, speaking, object detection, sound recognition, social interactions, and business transactions. Various techniques, including machine learning (ML), deep learning (DL), recommendation systems, text mining, predictive and prescriptive analytics, natural language processing, and predictive analytics, can be used to carry out tasks.

Cyber security can be addressed from two sides in the terms of AI; AI can be a tool to optimize the cyber security solutions or AI systems can be exploited and need cyber security solutions to be protected (Li, 2018).

Machine Learning is an approach to AI creation. It gives machines access to a large number of sample data and codes them to find patterns and make them learn on their own how to perform the task, rather than programming them by hand-coding software routines with a specific set of instructions to accomplish a particular task.

There are multiple applications of AI and machine learning in cyber security, such as *AI2*; an artificial intelligence platform to predict cyber-attacks that have been developed by MIT Computer Science and Artificial Intelligence Laboratory (CSAIL) and PatternEx, *CylanceProtec*t, which is an integrated information security threat prevention tool, which combines the benefits of artificial intelligence with information security controls to prevent malware infections, Darktrace, which is an information security solution, that can help detect and recognize emerging cyber threats that are able to avoid traditional information security protections, *Amazon Macie* that artificial intelligence provides tools for Macie to find, classify and protect sensitive data on Amazon Web Services (AWS), *Deep Instinct*, which is designed to protect organization’s mobile devices and services against known and unknown

malicious attacks in real-time or threat intelligence solutions like *IBM QRadar Advisor* and so on (Vähäkainu & Lehto, 2019).

Natural Language Processing (NLP) is the set of methods for making the human language accessible to computers (Eisenstein, 2018). NLP is a subfield of computer science, linguistics, and artificial intelligence. NLP is the area of AI-based deep learning that deals with the use of natural language in communication between people and machines. NLP offers a wide range of capabilities to improve human performance. NLP in risk and compliance may find standards and framework overlaps, data from the tech stack of a company, and threat feeds to find security flaws in your infrastructure. The ultimate goal of NLP is to "read," interpret, and comprehend language which is useful to the end-user. Contemporary approaches to natural language processing rely heavily on machine learning, which makes it possible to build complex computer programs from examples (Eisenstein, 2018).

The relationship between AI, NLP, ML, DL, and Linguistics can be seen in Figure 5 (Banerjee, 2020).

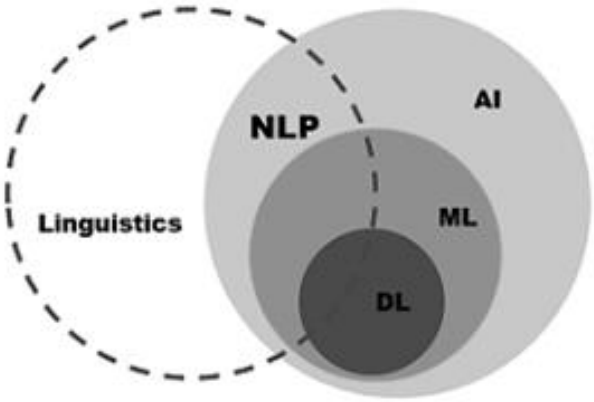


Figure 5: Relationship between AI, NLP, ML, DL, and Linguistics

Cybersecurity experts use many automated tools and technologies based on NLP to find, test, and correct weaknesses in a company's infrastructure, monitor malicious content on the system and identify network vulnerabilities (Ukwen & Karabatak, 2021).

There were multiple pieces of research on NLP and its use in the field of information security, such as the inventive study that identifies NLP use cases and applies theoretical and empirical ontological semantics (Atallah et al., 2001) or research that argues why NLP should move to information security and assurance (Raskin et al., 2002).

There are also multiple applications of natural language processing methods to be used for cyber security solutions; such as developing a multi-level ransomware detection framework (Poudyal & Dasgupta, 2020), and detecting social engineering attacks (Lansley et al., 2020). There were numerous other

studies on different areas of cyber security and natural language processing such as information security and assurance, information retrieval, forensics, file fragment classification, semantic knowledge representation, document clustering, intrusion detection, malware detection, malicious URL detection, phishing attack detection, ransomware detection, threat intelligence, DDoS (Distributed Denial of Service) attacks detection, privacy-preserving, vulnerabilities detection and operation and log anomaly detection (Ukwen & Karabatak, 2021).

There are also studies specifically focusing on the Turkish language to detect phishing attacks by URL (Buber et al., 2017), detecting Turkish phishing attacks with machine learning classifiers (Turhanlar, 2019) or filtering Turkish spam using LSTM techniques (Eryilmaz et al., 2020).

In e-mail spam filtering, modern machine learning and natural language processing algorithms for spam filtering, as well as approaches for assessing and contrasting various filtering techniques, were reviewed (Blanzieri & Bryl, 2008) and there is a study on the effectiveness of distinguishing spam from non-spam e-mails using word embedding and a pre-trained deep learning model (BERT) (AbdulNabi & Yaseen, 2021) or comparison between deep learning methods with traditional machine learning algorithms applied to spam e-mail detection by utilizing text representations from NLP (Srinivasan et al., 2020). And there was PhishNet-NLP for spam filtering, which combines context information with natural language processing techniques (Verma et al., 2012).

We will use natural language processing as a tool for the spam protection process as well.

2.3. Spam Filtering

Even though an absolute solution to spam e-mails is yet to be found, filtering methods are being evolved day by day to fight against spam. Spam filters are created to identify incoming dangerous e-mails from attackers or unsolicited marketers. Spam filters abundantly block lots of spam e-mails but it remains a weight to the networks, e-mail servers, and overall internet. They used to work rule-based and they were only following a checklist. As the technology evolved, spammers also evolved their techniques and rule-based-only filters became incapable. The rise in the volume of spam e-mails has created an intense need for the development of more dependable and strong spam filters. Hence artificial intelligence-based methods have been asserted to improve the filtering process, it is also observed in conjunction with the rule-based filtering methods. In artificial intelligence-based methods, spam detection is mostly performed using machine learning algorithms and deep learning techniques (Karim et al., 2019).

Even though the first spam message was delivered in 1978, it was not until 1982 for it to be recognized as a problem in the academic literature (Denning, 1982). The Bayes method, initially employed by Sahami et al in 1996 and

thereafter by other academics (Gabber et al., 1998; Hall, 1998; Sahami, 1996; Sahami et al., 1998) is the first mathematical tool applied to spam filtering systems. The foundation of Bayes' classifier is the well-known Bayes theorem, on which the first articles may be encountered as early as 1960 (Fisher, 1960).

For text classification, several machine learning techniques have been used (Apte et al., 1994; Dagan et al., 1997; Lewis et al., 1996). After being trained on manually categorized documents, these algorithms learn to categorize texts into predetermined categories based on their content. These kinds of algorithms have also been used to group e-mails into folders, thread e-mails (Lewis & Knowles, 1997), find relevant news articles (Faiz, 2006) and more. An attempt at using a machine learning algorithm for anti-spam filtering has been made by Sahami (Sahami et al., 1998). Sahami et al. reported outstanding precision and recall on unseen messages after training a Naïve Bayesian classifier (Duda & Hart, 1973) on manually classified valid and spam messages. It may come as a surprise that text classification can be useful in anti-spam filtering since, unlike other text categorization tasks, the act of mass mailing a message without reading it first, rather than its content, constitutes spam. However, it appears that spam's language is a different genre, and since spam communications frequently discuss subjects which are not covered in valid messages, it may be able to train a text classifier for anti-spam filtering.

Naïve Bayes Classifier has been utilized to solve a wide range of problems, from the classification of texts in news organizations to the initial diagnosis of illnesses in medicine. The presence or lack of words in the text is typically chosen as a characteristic for the situations where Naïve Bayes Classifier is applied. In the case of e-mail filters that classify messages as spam, the header (fields holding generic message details such as the subject, sender, and receiver), topic, and body (the actual contents of the message) of the e-mail are all taken into consideration.

Then the method of overlapping probability proposed by R. Fisher in 1950 as in Gary Robinson's paper (Robinson, 2003) was used for filtering purposes in a statistical approach. Robinson offered to determine both the likelihood that an e-mail is "legitimate" and its likelihood of being spam in order to detect spam. After this, some other works addressed the use of Markov chain PageRank (Boldi et al., 2005) and Hidden Markov Model were the following directions (Gordillo & Conde, 2007). A brand-new technique for digitally analysing textual e-mails for spam identification can also be seen in the literature (Korelov et al., 2006). In some following works, the application of clustering analysis techniques to the issue of separating authentic e-mails from spam is discussed (Hsiao & Chang, 2008; Lee et al., 2010).

In an evolutionary situation made possible by the usage of filters (Goodman et al., 2007), spammers use tools (Stern, 2008) with a variety of strategies designed particularly to reduce the number of messages which are detected.

The earlier types of spam filtering techniques are created to identify incoming dangerous e-mails from attackers or unsolicited marketers. Since spammers constantly change external signs of e-mails to skip spam filtering systems, there arises a need for an adaptive filtering system, which should have the ability to react quickly to the changes and provide fast and qualitative self-tuning. The course of action to be done once they have been located typically depends on the filter's application setting. They are typically delivered to a folder which only contains messages tagged as spam if used by a single user as a client-side filter, making it easy to identify these messages. A mail server's filter, on the other hand, may handle messages from many users and either mark them as spam or remove them. Another potential is a collaborative environment, where filters operating on several machines share knowledge of the messages they have received in order to function better.

The rise in the volume of spam e-mails and the change in the techniques of spammers have created an intense need for the development of more dependable and strong spam filters. Since 2009, starting with the publication of Cortez et al. (Cortez et al., 2009) the claim that symbiotic data mining is a combination of collaborative filtering and content-based filtering has been coming to reality.

Automatic filtering rules and e-mail categorization utilizing machine learning techniques like Naïve Bayesian classification, Support Vector Machine, K Nearest Neighbour, and Neural Networks are typically created using Content-Based Filtering. In order to filter incoming e-mail spam, this technology often analyses terms, the incidence, and distribution of words and phrases in e-mail content (Christina et al., 2010).

There is Previous Likeness Based Spam Filtering Technique; it classifies incoming e-mails based on how closely they resemble stored examples using memory-based, or instance-based, machine learning techniques (e.g. training e-mails). A multi-dimensional space vector is created using the e-mail's properties, and new instances are plotted as points using this vector. The most well-liked class of its K-closest training instances is then given the fresh instances (Sakkis et al., 2001). It filters spam e-mails using the k-nearest neighbour (kNN) algorithm (Mucherino et al., 2009).

Another method is Adaptive Spam Filtering Technique. This method classifies spam into distinct categories in order to detect and filter it. It separates an e-mail corpus into different groups, each has a unique text. Each incoming e-mail is compared to each group, and a percentage of similarity is calculated to determine the most likely group to which it belongs (Pelletier et al., 2004).

There is also Heuristic or Rule-Based Spam Filtering Technique, which compares a large number of patterns, most of which are regular expressions, against a selected message using pre-made rules or heuristics. A message's grade is raised when there are several related patterns. If any of the patterns

did not match, it subtracts from the score. Any communication that receives a score beyond a certain level is classified as spam; otherwise, it is considered to be authentic. While certain ranking criteria do not vary over time, others need to be updated often in order to successfully combat the threat of spammers who constantly add new spam messages which can easily evade detection by e-mail filters (Christina et al., 2010). SpamAssasin is a good example of a rule-based spam filter (Mendez et al., 2006).

One of the widely used spam filtering techniques is Case Base Filtering. First, using a collection approach, all e-mails spam and non-spam/ham are collected from each user's e-mail. Then, utilizing the client interface, feature extraction, selection, grouping of e-mail data, and process evaluation, pre-processing stages are carried out to change the e-mail. After that, the information is divided into two vector sets. The machine learning approach is also used to test and train datasets to determine if incoming e-mails are spam or not (Christina et al., 2010).

Different e-mail spam classification approaches have been proposed by numerous researchers and academics, and they have been utilized successfully to divide data into groups.

The most successful technique applied in filtering spam is the content-based spam filtering approach which classifies e-mails as either spam or ham depending on the data that made up the content of the message. Bayesian filtering, SVM, kNN classifier, neural networks, AdaBoost classifier, and other methods are examples of this methodology. Systems based on the machine learning approach facilitate learning and adjustment to recent dangers posed to the security of spam filters. They also have the capacity to counter curative channels spammers are using (Dada et al., 2019).

Before raising our research questions about Turkish spam data, we scanned the literature for Turkish e-mail datasets for spam filtering. One study defines adaptive anti-spam filtering for Turkish (Özgür, 2003), also there were time-efficient methods on Turkish spam data (Çiltik & Güngör, 2006), and low time complexity methods were also studied. The dataset is formed from the messages of one of the authors since there was no dataset for Turkish, and 640 ham, and 640 spam messages were used for balance (Güngör & Çiltik, 2007).

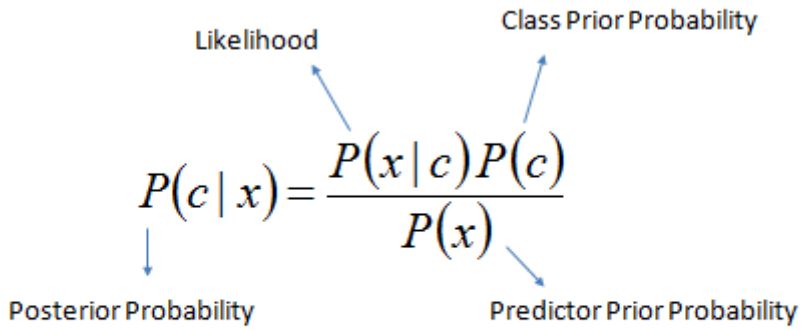
There were also studies on the classification of spam with different methods, like the artificial immune system. The dataset in the study was created from spam sent to the contact e-mail address added to the homepage of the official website of Siirt University, and Turkish e-mails sent to personal e-mail addresses. There are 603 regular and 540 spam mails in total (Özdemir et al., 2013). The e-mail dataset to be used for spam filtering had to be recollected and processed than in most studies.

Text classification categorizes the raw text into a group of words. It allows us to label the unstructured texts with their relevant tags which are predicted from a set of predefined categories. Using NLP, text classification can automatically analyse text and then assign a set of predefined categories or tags based upon its context. NLP can be used for topic detection, sentiment analysis, and language detection. One of the most common uses of text classification is spam filtering and it is widely used by service providers. E-mail spam filtering is an important issue in network security and it has also become important in machine learning techniques. Several machine learning algorithms have been employed for e-mail spam filtering, including algorithms which are considered top-performers in Text Classification (Rathi & Pareek, 2013) like Boosting algorithm, Support Vector Machines (SVM) algorithm (Kumar et al., 2016) and Naïve Bayes algorithm (Feng et al., 2016) XGBOOST (Chen & Guestrin, 2016) and K Nearest Neighbour (KNN) (Mucherino et al., 2009).

Naïve Bayes classifier that uses Naïve Bayes algorithm to classify has a very important role in the process of filtering spam e-mail. The quality of performance Naïve Bayes classifier is also based on datasets. In the research done by Nurul Fitriah Rusland et al in 2017, it can be seen a dataset that has fewer instances of e-mails and attributes can give good performance for Naïve Bayes classifier (Rusland et al., 2017).

Naïve Bayes is a classification technique based on Bayes' Theorem with a conjecture of independence amid predictors (Bayes, 1763). Bayes' Theorem is simply a mathematical formula which is used for calculating conditional probabilities. Conditional probability is a measure of the probability of something occurring given that another thing has occurred.

The formula of Bayes' Theorem as in Figure 6, tells us how often c happens, given that x happens, written P(c|x) also called posterior probability, when we know: how often x happens given that c happens, written P(c|x) and how likely x is on its own, written P(x) and how likely x is on its own, written P(x).



$$P(c \mid X) = P(x_1 \mid c) \times P(x_2 \mid c) \times \cdots \times P(x_n \mid c) \times P(c)$$

Figure 6: Bayes Theorem Formula

Posterior Probability: Probability of c occurring given evidence of x already occurred.

Likelihood: Probability of x occurring given evidence of c already occurred.

Class Prior Probability: Probability of c happening.

Predictor Prior Probability: Probability of x happening.

In order for the algorithm to calculate the likelihood that a text belongs to a category, any vector used to represent a text must include information about the probabilities that particular words will appear in texts belonging to that category. Since it is based on independent probabilities; it is possible to obtain successful results, even when there are limited computational resources and a small dataset.

Support Vector Machines (SVM), which, like Naïve Bayes, requires little training data to begin producing reliable results. But SVM needs more computing power than Naïve Bayes. In essence, SVM creates a "hyperplane" or line that separates a region into two subspaces. Vectors (tags) that are members of one group are found in one subspace, whereas those that are not members of that group are found in another subspace. The hyperplane with the greatest distance between each tag is the ideal hyperplane.

The basic workflow of text classification in machine learning can be seen in Figure 7.

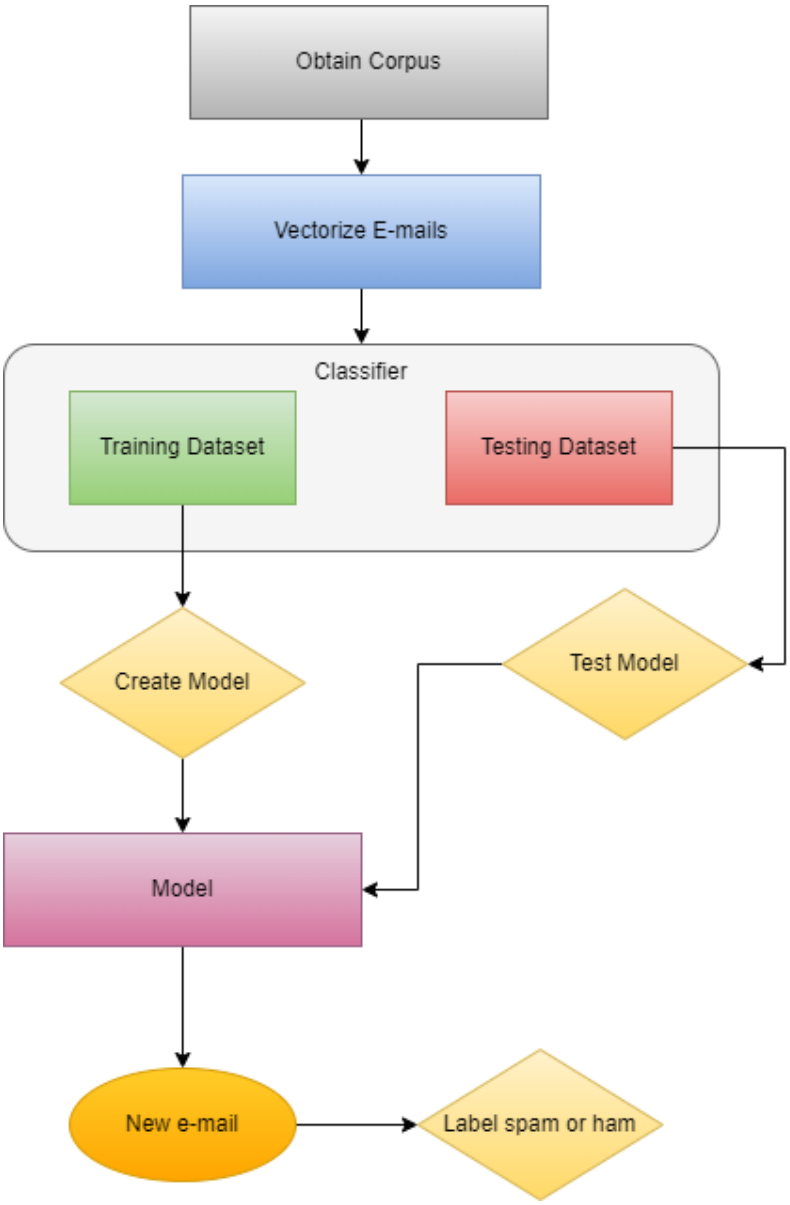


Figure 7: Text classification task on spam filtering

Since spam filtering is a text classification problem in the machine learning approach, it requires a classifier to label the e-mail data from the input dataset as ham or spam.

In machine learning problems, the quality and the quantity of data are crucial and for these reasons we come back to our research question, to be able to get more qualified data from a considerably small set of data would be the solution for some machine learning cases.

In practice, the training step can be replaced by using a pre-trained model which is applicable to the machine learning problem at hand.

Machine learning algorithms are usually grouped by their learning method as follows; Supervised Machine Learning, Unsupervised Machine Learning, Semi-Supervised Machine Learning, and Reinforcement Machine Learning. These groups are formed by how the algorithm can model a problem based on its interaction with the input data which can also be called environment or experience. In our study, we will be using supervised machine learning methods due to our problem to solve.

2.4. Language Modelling and Augmentation in Natural Language Processing

Natural Language Processing (NLP) is the set of methods for making the human language accessible to computers (Eisenstein, 2018). Contemporary natural language processing techniques mainly rely on machine learning models which enable the creation of complicated computer programs and machine learning models need some kind of quantitative representation for their calculations in order to process words. Developing meaningful representations of text has been one of the primary goals of NLP since its start. Text representation is the process of representing the text as numbers so that computers can work on them. Depending on the problem to be resolved, the text in question can be a document, a sentence, or a word.

Text representations in NLP systems are applied in increasingly flexible and task-agnostic ways for downstream transfer. First, single-layer representations such as word2vec and GloVe were learned using word vectors (Mikolov et al., 2013; Pennington et al., 2014) and fed to task-specific architectures, then RNNs with multiple layers of representations and contextual state were used to form stronger representations (Dai & Le, 2015; Peters et al., 2018) which are applied to task-specific architectures, and more recently pre-trained recurrent or transformer language models (Vaswani et al., 2017) have been directly fine-tuned, entirely removing the need for task-specific architectures (Brown et al., 2020).

We will be mentioning some of the techniques that are relevant to our work.

2.4.1. Frequency-based Representations

One-hot-encoding: One hot encoding is a process by which categorical variables are converted into a form that could be provided to machine learning algorithms to do better predictions. In one hot encoding, while words can be easily digitized, when two different words are given, it is not possible to extract the relationships between these two words. Since there are as many

vector sizes as the number of words, there are too many "0"s in the vector. It requires a lot of memory.

Count vector: They are vectors created according to the frequency of words in the document.
TF-IDF: TF-IDF takes into account not only the rate of occurrence of words in the document but moreover the frequency of occurrence of a word in other documents (Ramos, 2003). If it is a word or phrase that appears frequently in all documents, the TF-IDF value will be low. Texts are converted into vectors using the bag of words technique in text categorization (Thorsten Joachims, 1998). Despite the fact that TF-IDF BoW (bag of words) representations assign weights to various words, the word meaning cannot be captured by them.
Co-occurrence: It is a matrix that holds the number of words appearing together. It requires large memory space and processing power.
Bag of Words: This method gives each word that appears in the text a special token, typically a number.

2.4.2. Word Embeddings

The fact that the vocabulary size grows together with the size of the vector representation of texts is a noticeable disadvantage of the methods mentioned, in addition to their inability to capture word semantics. As a result, a vector with many zero scores is produced. This vector is referred to as a sparse vector or sparse representation, and it requires more memory and processing resources during modelling. Word embeddings use dense representations to reduce dimensionality and use contextual similarity to offer a more expressive representation.

Word Vectors: Word embedding methods learn real-valued vector representations for a predetermined fixed-size vocabulary which has been obtained based on a text.

The learning process can work in some parts in combination with a neural network model (text classification) or as an unsupervised process using document statistics.

Cosine Similarity: Cosine similarity measures the similarity between two vectors of an inner product space. It is measured by the cosine of the angle between two vectors and determines whether two vectors are pointing in roughly the same direction. It is often used to measure document similarity in text analysis (Han et al., 2012).

2.4.2.1. Similarity-based Word Embedding Methods

Word2vec

Word2vec is an unsupervised and prediction-based model which represents words in vector space. It is an unsupervised learning technique to learn continuous representations of words that have one input, one secret layer and one output. It is developed by Google researcher Tomas Mikolov and his team in 2013 (Mikolov et al., 2013). The biggest advantage of the Word2vec model is that the closeness between the words is not lost due to the conditional probability principle working logic according to the positions of the words in the sentence.

CBOW (Continuous Bag of Words) and Skip-gram

Word2vec works in two ways; CBOW (Continuous Bag of Words) and Skip-gram. We can see Figure 8 to understand CBOW and Skip-gram (Mikolov et al., 2013).

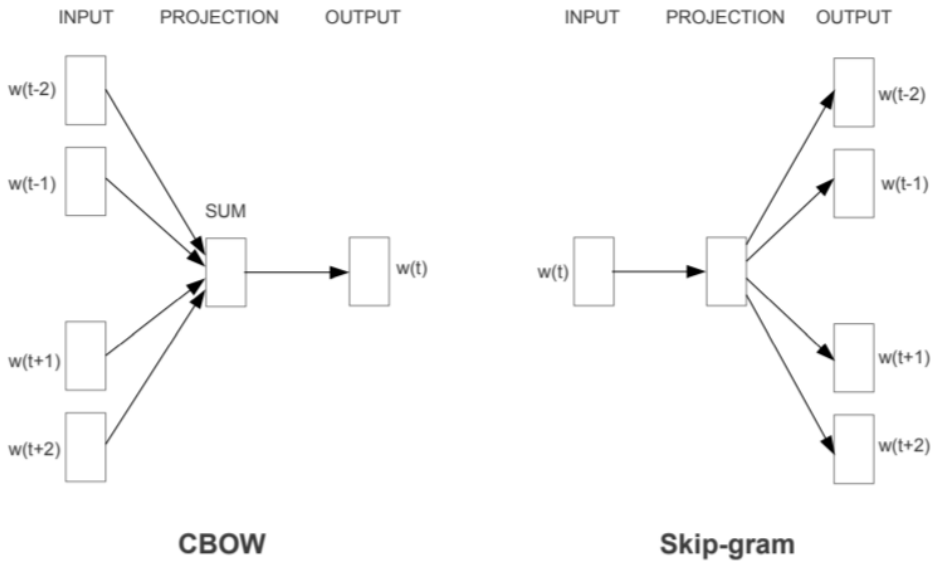


Figure 8: The CBOW and Skip-gram models

These two methods generally work similarly but they have different kinds of inputs and outputs. When creating word vectors, there are some hyperparameters such as “window size” or embedding size. Window size remarks how many words there must be around the target word and embedding size indicates how many dimensions of a vector each word has to be defined with. This also corresponds to the number of neurons on the secret layer.

In the CBOW model, the words not in the centre of the window size are taken as input and the words in the centre are tried to be predicted as output, while in the Skip-gram model, the word in the centre is taken as input and the words which are not in the centre are tried to be predicted as output. This process continues until the sentence ends. These operations are applied to all sentences and thus, mapping is applied to the unlabelled data we have at the beginning and then it becomes ready to train.

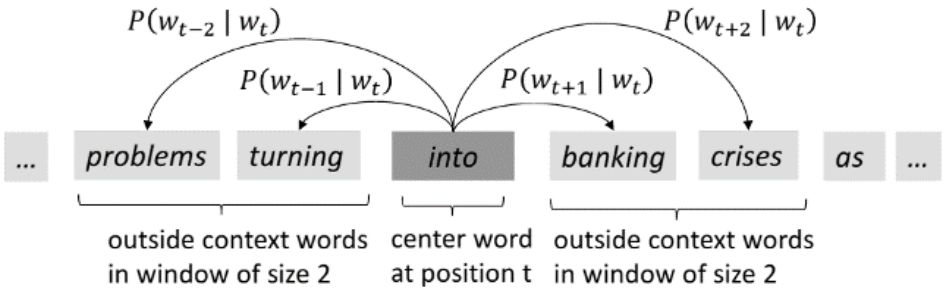


Figure 9: Word2vec Mechanism

The hidden state shown in red in Figure 9 can maintain information about the previous word in the sentence.

FastText

FastText is an extension of Word2vec and a library in the gensim structure developed by Facebook AI Research team (Joulin et al., 2017) . Instead of inputting individual words to the neural network, fastText splits words into "n-grams" based on several letters.

For example, for the word apple - “elma”, the tri-grams should be: “elm” and “lma”. In the n-gram expression, n represents the number of repeats. In other words, the n expression here provides how many times the word would be divided. It allows us to understand how much of a word or letter.

In fastText, some words can be expressed using n-grams, even though they are not in the training dataset. Since the number of n-grams will be many times higher than the number of words, the training period is also extended. On the other hand, it can express words which are found in small numbers in documents better than Word2vec, it is stated that it is faster than other methods and it supports Skip-gram with CBOW.

There are pre-trained fastText models for many more languages than any other embedding algorithm since it needs less data than others for training.

Glove

GloVe, The Global Vectors for Word Representation, is a word2vec extension developed by Stanford as an open source (Pennington et al., 2014).

The unsupervised algorithms of Glove are based on the statistics of the data. Models such as Skip-gram and CBOW capture semantic information but do not use collaborative statistics. Although matrix parsing methods use these statistics, they cannot capture semantic relationships. The "GloVe" model aims to solve this problem by creating a new objective function using probability statistics. The GloVe model's fundamental principle is to embed words in meaningful vectors by concentrating on the likelihood that words will occur together in a corpus of texts. In other words, GloVe examines the frequency with which two terms are used together throughout the whole corpus of texts.

2.4.2.2. Transformer

The idea of training a different language model to create better contextual word representation has been very effective in many NLP tasks, however, they have their disadvantages such as being difficult to train or parallelize, having short-term memory etc. The Attention mechanism introduced in the work by Vaswani, A. et al. in 2017 aimed to overcome most of these problems. The other NLP models usually capture all the information in the input sentence -the details of objects, how objects are related to each other etc. in an intermediate state and then use this intermediary information and express it in the output. The size of the vector used for this intermediary state before starting to decode the output sequence is fixed. In this entire process, the intermediate stage is crucial. The accuracy of the output from the decoding process depends on its capacity to retain all the data passed in the input sentence. The most important component is still the intermediate state. In cases with very long texts as input, the intermediate state fails and is not sufficient to capture all the information. The idea of attention enhances the model's performance and frees the intermediate state from being entirely in charge of encoding all the information accessible to the decoder in a fixed length vector and from being a potential bottleneck. The information may be distributed throughout the sequence of annotations-encoder hidden states, and the decoder may recover it selectively as necessary. A transformer model can "attend" or "focus" on all previous tokens that have been generated (Vaswani et al., 2017).

Transformers are semi-supervised machine learning models that are primarily used with text data and have replaced recurrent neural networks in natural language processing tasks. The encoder maps an input sequence of symbol representations $(x_1, ..., x_n)$ to a sequence of continuous representations $z = (z_1, ..., z_n)$. Given z, the decoder then generates an output sequence $(y_1, ..., y_m)$ of symbols one element at a time. At each step, the model is auto-regressive

consuming the previously generated symbols as additional input when generating the next (Vaswani et al., 2017).

The Transformer follows the architecture using stacked self-attention and point-wise, fully connected layers for both the encoder and decoder, shown in the left and right halves of Figure 10 respectively (Vaswani et al., 2017).

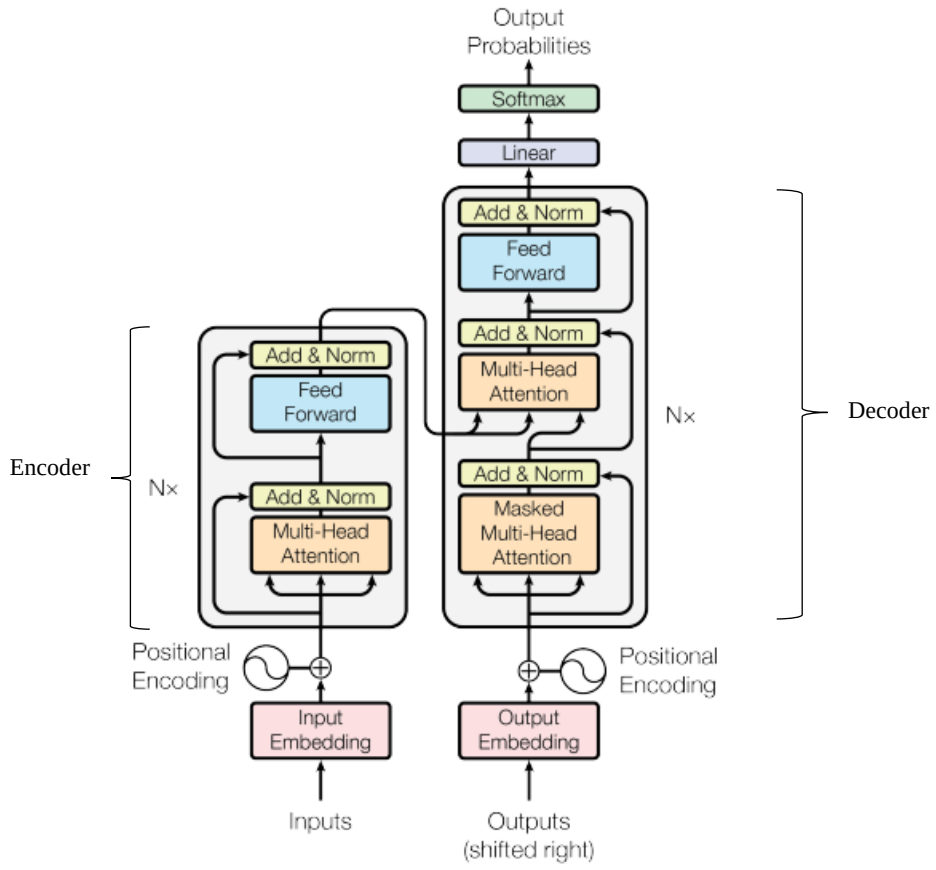


Figure 10: The Transformer model architecture

Encoder: The encoder maps an input sequence of symbol representations (x_1, …, x_n) to a sequence of representations $z = (z_1, …, z_n)$.

Decoder: Given z, the decoder generates an output sequence (y_1, …, y_m) of symbols one element at a time.

GPT

The Generative Pre-Trained Transformer (GPT), is created by OpenAI. It is an unsupervised generative model, which means it attempts to produce an appropriate response from an input such as a sentence and the training data were not labelled. The GPT-2 language model, developed by OpenAI in February 2019, uses unsupervised deep learning transformers to predict the next word or words in a sentence (Radford et al., 2019). GPT-2 can learn language skills such as reading, summarizing, and translating an unstructured text without the use of domain-specific training data. GPT-3 was developed to be more resilient than GPT-2; it can handle a wider range of specialised themes. A more accurate description may be that it is a sequential text prediction model, even if it is still a language prediction model.

BERT

BERT, stands for Bidirectional Encoder Representations from Transformers, was proposed in 2018 as a new language model representation. Unlike recent language representation models, BERT is designed to pre-train deep bidirectional representations from the unlabelled text by jointly conditioning on both left and right contexts in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications (Devlin et al., 2018).

In BERT, from output layers, the same architectures are used in both pre-training and fine-tuning. The same pre-trained model parameters are used to initialize models for different downstream tasks. During fine-tuning, all parameters are fine-tuned. [CLS] is a special symbol added before every input example, and [SEP] is a unique separator token.

In Figure 11, the pre-training and fine-tuning procedures of BERT can be seen (Devlin et al., 2018).

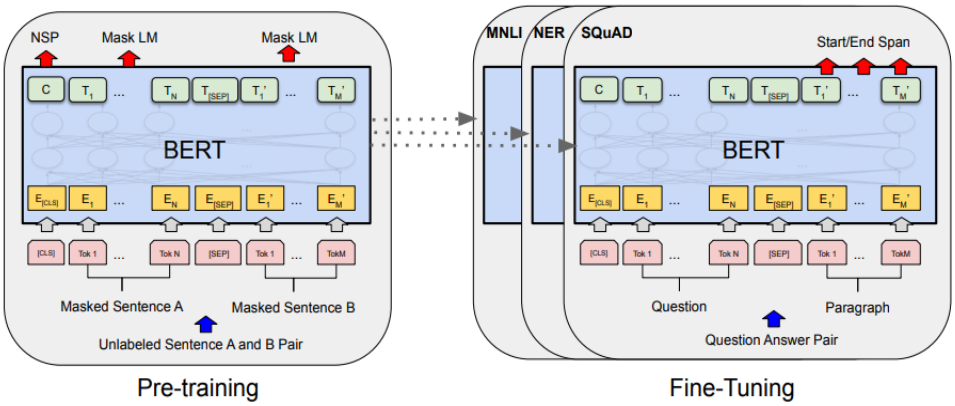


Figure 11:Overall pre-training and fine-tuning procedures for BERT

There are two tasks are introduced for pre-training with BERT; Masked Language Model (MLM), Next Sentence Prediction (NSP) and the results for 11 NLP tasks are presented and claim these results enable even low-resource tasks to benefit from deep unidirectional architectures (Devlin et al., 2018).

DistilBERT

Knowledge distillation, also known as teacher-student learning, is a compression approach in which a small model is educated to mimic the behaviour of a bigger model (or a group of models) wihch is introduced by Bucilă, C. et al., and generalized by Hinton, G. et al. several years later (Bucilă et al., 2006; Hinton et al., 2015).

DistilBERT is the distilled version of BERT, it has the same general architecture as BERT and it is a BERT base-trained Transformer model that is compact, quick, affordable, and light. Over 95% of BERT's performance on the GLUE (The General Language Understanding Evaluation) language understanding benchmark is preserved despite having 40% fewer parameters and running 60% faster than BERT-base-uncased (Sanh et al., 2019; Wang et al., 2019). The token-type embeddings and the pooler are removed while the number of layers is reduced by a factor of 2. Most of the operations used in the Transformer architecture (linear layer and layer normalisation) are highly optimized in modern linear algebra frameworks and the investigations show that variations on the last dimension of the tensor (hidden size dimension) have a smaller impact on computation efficiency (for a fixed parameter

budget) than variations on other factors like the number of layers. Thus it focuses on reducing the number of layers.

Model	**Score**	CoLA	MNLI	MRPC	QNLI	QQP	RTE	SST-2	STS-B	WNLI
ELMo	68.7	44.1	68.6	76.6	71.1	86.2	53.4	91.5	70.4	56.3
BERT-base	79.5	56.3	86.7	88.6	91.8	89.6	69.3	92.7	89.0	53.5
DistilBERT	77.0	51.3	82.2	87.5	89.2	88.5	59.9	91.3	86.9	56.3

Figure 12: The comparison of models

In Figure 12, the performance comparison of the models can be seen (Sanh et al., 2019).

CHAPTER 3

METHODOLOGY

In this chapter, the proposed augmentation and the classification models will be explained. For the first phase, we needed to find a suitable way to augment an e-mail dataset in Turkish. Therefore, our first step was finding or collecting the appropriate data for our study. After acquiring the dataset, we applied the relevant pre-processing methods to the data and implemented different augmentation methods on the dataset. We compared the classification results before and after the augmentation with the selected methods and presented the comparison of them in the relevant section. In all stages of this study, Python 3 was used as the programming language. For the environment, we started with Jupyter Notebook but then finalize most of the work with Google Colab.

We went through multiple ways throughout the study to find the most suitable methods to answer our research questions. First, we have started with word2vec methods along with gensim library to replace words in sentences according to their cosine similarity. Then, we have done an implementation of Transformer GPT-2 model that generates data. Finally, we have moved on to the BERT and distilBERT of Transformer with specific augmentation libraries. Subsequently, we used different techniques which can be used as features in order to augment text data in Turkish language by similarity. We presented the frameworks we used for each method. After improving on these building blocks, we finally presented the augmented data and moved on to the second phase. In our second phase, we performed text classification, spam filtering in our specific case. We showed the accuracy of the spam classification results of the initial dataset and the augmented dataset. The details of the processes of each method are explained in their relative sections.

3.1. Obtaining Data

First, we tried to create our own dataset for this study. For this purpose, we got an unstructured e-mail dataset from METU IT Department. This dataset was collected from automated filters and hand-written protocols. However, since this dataset had so many types of data, many false positives, and e-mails in different languages, mostly English, it took some time to get usable data from it. After selecting the suitable e-mails that are in Turkish, the dataset became significantly small. We had to label the data as spam and ham manually. Some e-mails were too repetitive since the data was collected from

multiple sources. At the beginning of the study, we used this dataset to train a model and then use it with a pre-trained model to create new e-mails according to word similarity but it gave poor results at every attempt. Then we tried to find other datasets from IT Departments of some institutions but it did not serve as a suitable solution due to data transformation processes since the e-mails might include sensitive information. We also considered translating English e-mails to Turkish but since the translation itself is also an NLP problem, we decided not to use this method and stay focused on our task. Finally, we decided to use a Turkish e-mail dataset from Kaggle which is updated in 2019 and includes a total of 330 spam and 496 ham e-mails which were collected from several personal accounts (Demir, 2019).

We used Python's pandas library to use the dataset inside the code, which was stored as a CSV document. We used the read_csv method of pandas library and turn our dataset into a dataframe object. Pandas dataframe is a two-dimensional, size-mutable, potentially heterogeneous tabular data structure in Python. It contains rows and columns and it is considered one of the most efficient ways to keep and represent CSV file information in Python code.

In Figure 13, we present the data, an example of an e-mail in the dataset, as it is stored in Kaggle.

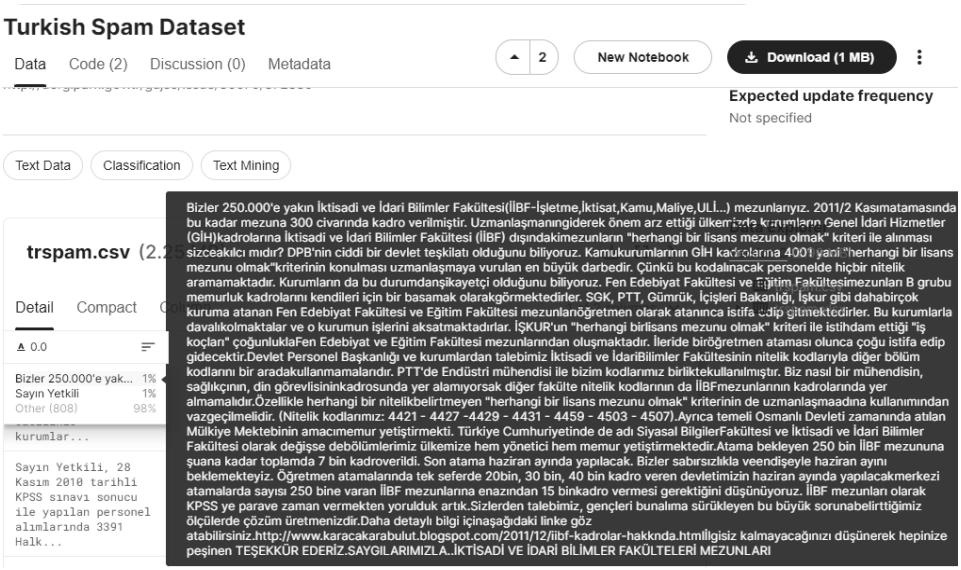


Figure 13: Example e-mail shown in source

In Figure 14, we can see the statistics of the same example e-mail in Figure 13.

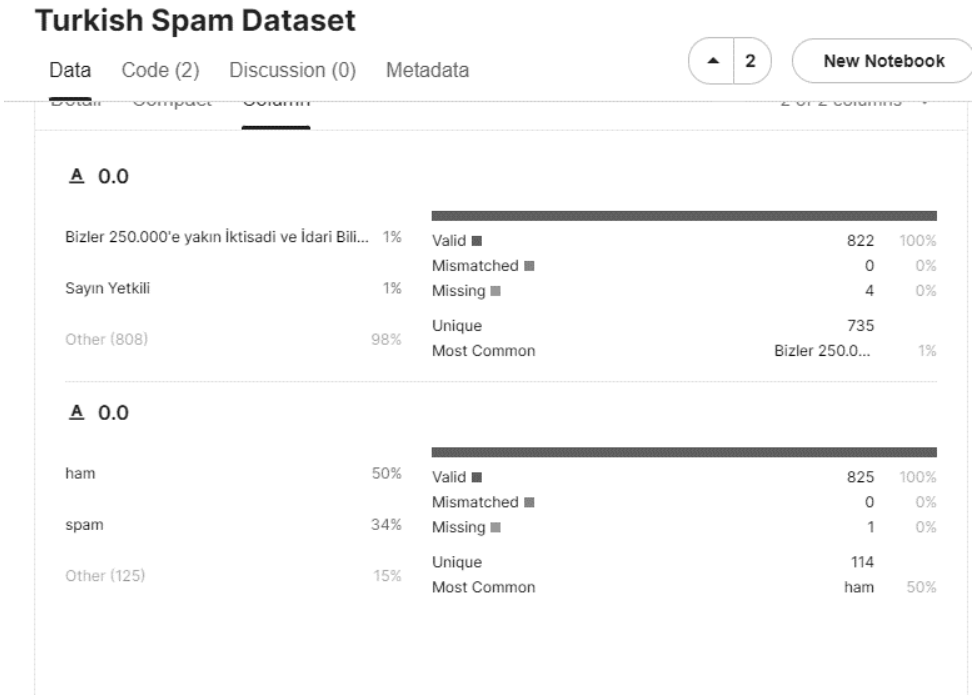


Figure 14: Statics of the example e-mail in source

Additionally, in Figure 15, a sample data inside our code can be seen as it is represented in a pandas dataframe.

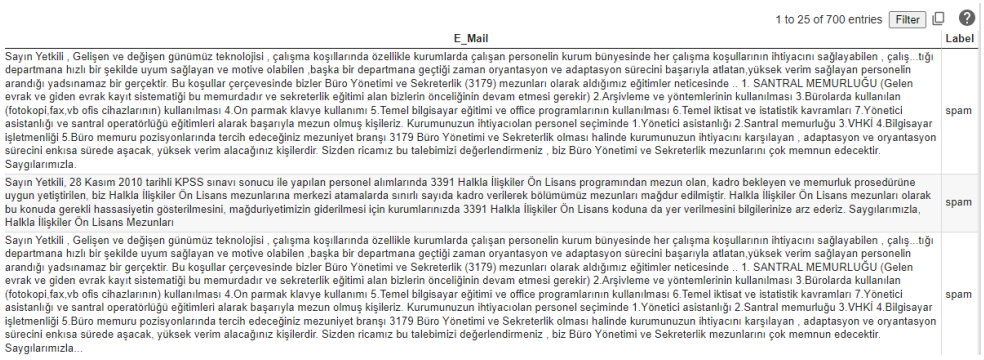


Figure 15: Original dataset shown as pandas dataframe

3.2. Preparing the Data

The data collected from a source is generally unstructured. It contains unusual text, meaningless characters or different languages mixed with each other. Therefore, raw texts mostly cannot fit in machine learning or deep learning models; the models cannot interpret the unusualness, so according to the problem to be solved, the text might need to be cleaned or pre-processed first.

The steps of the pre-processing differed according to the model we chose. Even though we will be explaining all the pre-processing details in this section, we will clarify the steps used for each method in their own sections. In the classification task, we have run the pre-processing steps for every method. All the pre-processing steps can be seen in Appendix A.

Primarily, we reshaped the dataset as it might had duplicate rows and columns. After removing duplicates, we had a 59.42% ham and 40.57% spam ratio of e-mails in our dataset.

Pre-processing Methods to Be Used Only in Turkish Dataset

The pre-processing steps we used had slight differences according to the methods. For example, we used every pre-processing step mentioned here for word2vec implementation so that the data can be used as a valuable input since word2vec needs the cleanest form of a word but we did not use some of the steps with BERT or GPT-2 implementations. In this section, we will explain all the pre-processing steps for Turkish and we will go into details of usage in the relevant sections.

In Turkish, we have some unique cases to be solved and we also need to do that in a certain order. To achieve that, we used some of the methods explained here before processing the data to observe the results. The upper-case lower case issue for Turkish is basically about the letter ‘i’ which has the ‘İ’ as the upper case for Turkish and ‘I’ for English. Because of this issue, default functions to turn letters into upper and lower forms in programming languages and libraries might give wrong results for Turkish. So it is solved with a small function instead.

After that, we remove digits, punctuations, HTML tags and white spaces since we want to focus on words and words represented as strings and these characters would be irrelevant to the study.

Since our data consists of e-mails and some of them are random e-mails to be sent as spam, there were a lot of irregularities. Some words that should be written in Turkish characters were written in English characters. For example, the word ‘akşam’ (evening) can be found as ‘aksam’ and both words are calculated as different words as in fact, they are not. This is called asciifying and it became a problem for us since we need the actual form of the word in the original language with Turkish characters. This problem can be solved with a library called Deasciifier (Sevinç et al., 2020). With this library, we

regularized most of the expressions which are written in ASCII characters even though they should not have been so.

Furthermore, we followed some pre-processing steps for Turkish text which are shown in the following Figure 16. These steps are implemented according to the needs of the methods used.

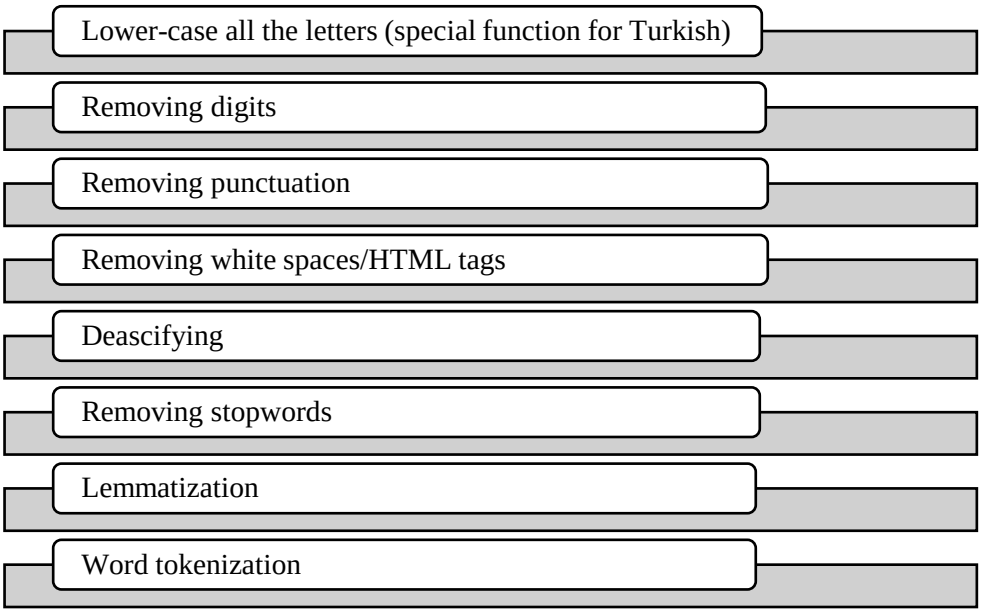


Figure 16: Pre-processing of Turkish Data

Stopwords

Sometimes, some widespread words which would appear to be of little value in helping select documents matching a user's need are excluded from the vocabulary entirely. These words are called stopwords (Manning et al., 2009a).

The stopwords for Turkish are "acaba, ama, aslında, az, bazı, belki, biri, birkaç, birşey, biz, bu, çok, çünkü, da, daha, de, defa, diye, eğer, en, gibi, hem, hep, hepsi, her, hiç, için, ile, ise, kez, ki, kim, mı, mu, mü, nasıl, ne, neden, nerde, nerede, nereye, niçin, niye, o, sanki, şey, siz, şu, tüm, ve, veya, ya, yani" as stated in the open source NLTK library.

Stopwords are words that do not add any unique meaning to sentences. Thus, they are usually removed for natural language processing tasks. We used NLTK library to remove stopwords. By removing stop words, we remove the low-level information from our text and this helps us focus on the important information in our data. We also removed words shorter than 2 (two) characters since these words in Turkish do not contribute to the meaning of the text.

Lemmatization

The goal of lemmatization is to reduce inflectional forms and sometimes derivationally related forms of a word to a common base form (Manning et al., 2009b). For grammatical reasons, different forms of words are used in most languages and this is the case for Turkish as well. For instance; the Turkish word "bataklığa", which can be found in a text like this, has suffixes and needs to be processed to acquire its root. The process is shown below.

1. Bataklığa (noun with suffix)
2. bataklık (noun)
3. batak (noun)
4. bat- (verb)

In Turkish, words might have some suffixes that change their characters at the end. Also, there are derivationally related words with similar meanings, we can see them in the example as "batak" and "bataklık".

Sometimes it would be useful to count these words as one and other times it might be better to separate them and then relate them to each other. The goal here is to reduce these variations of words to a common base form. Lemmatization usually uses vocabulary and morphological analysis of words and aims to return the base or dictionary form of a word called "lemma".

There is also a process which is called stemming to find the root of words but since it does not use morphological forms of the word, it is not very applicable to our study.

There are multiple tools for English lemmatization but we used the rich Zemberek library that is specified for Turkish and we got the lemmas of the words via Zemberek (Akın & Akın, 2007).

Table 1: Lemmatization Examples

Input Word	**Output Word**
gelişen (improving)	geliş (to improve)
rehberlik (guidence)	rehber (guide)
mesleğe (to a profession)	meslek (profession)
saygılarımızla (with our respect/best regards)	saygı (respect)
yönetici (manager)	yönet (to manage)
asistanlığı (assistantship of)	asistan (assistant)

Some examples of the results of lemmatizations are given in Table 1.

In Figure 17 we can see an example of pre-processing and lemmatization conducted on an e-mail in our dataset.

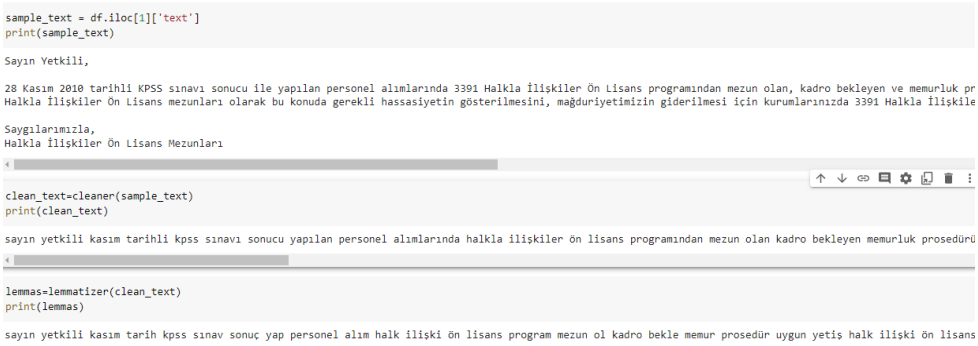


Figure 17: An example e-mail of pre-processing and lemmatization conducted

Table 2 shows us the results of a lemmatization process of an example e-mail text from our dataset.

Table 2: Pre-processing and Lemmatization Results

Original Text
Sayın Yetkili, 28 Kasım 2010 tarihli KPSS sınavı sonucu ile yapılan personel alımlarında 3391 Halkla İlişkiler Ön Lisans programından mezun olan, kadro bekleyen ve memurluk prosedürüne uygun yetiştirilen, biz Halkla İlişkiler Ön Lisans mezunlarına merkezi atamalarda sınırlı sayıda kadro verilerek bölümümüz mezunları mağdur edilmiştir. Halkla İlişkiler Ön Lisans mezunları olarak bu konuda gerekli hassasiyetin gösterilmesini, mağduriyetimizin giderilmesi için kurumlarınızda 3391 Halkla İlişkiler Ön Lisans koduna da yer verilmesini bilgilerinize arz ederiz. Saygılarımızla, Halkla İlişkiler Ön Lisans Mezunları

Table 2 (cont.)

After The First Cleaning
sayın yetkili kasım tarihli kpss sınavı sonucu yapılan personel alımlarında halkla ilişkiler ön lisans programından mezun olan kadro bekleyen memurluk prosedürüne uygun yetiştirilen halkla ilişkiler ön lisans mezunlarına merkezi atamalarda sınırlı sayıda kadro verilerek bölümümüz mezunları mağdur edilmiştir halkla ilişkiler ön lisans mezunları olarak konuda gerekli hassasiyetin gösterilmesini mağduriyetimizin giderilmesi kurumlarınızda halkla ilişkiler ön lisans koduna yer verilmesini bilgilerinize arz ederiz saygılarımızla halkla ilişkiler ön lisans mezunları
After Lemmatization
sayın yetkili kasım tarih kpss sınav sonuç yap personel alım halk ilişki ön lisans program mezun ol kadro bekle memur prosedür uygun yetiş halk ilişki ön lisans mezun merkez ata sınır sayı kadro ver bölüm mezun mağdur et halk ilişki ön lisans mezun ol konu gerekli hassasiyet göster mağduriyet gider kurum halk ilişki ön lisans kod yer ver bilgi arz et saygı halk ilişki ön lisans mezun

Tokenization

Given a character sequence and a defined document unit, tokenization is the task of chopping it up into pieces, called tokens, perhaps at the same time throwing away certain characters, such as punctuation (Manning et al., 2009a).

Tokenization is the process where we put the input and split it into pieces which will be meaningful for the process it will be used. For our case, we could use sentence tokenization and word tokenization. And for word tokenization, we used NLTK word tokenizer, a specialized Turkish word tokenizer library and also Regex. They all worked in some cases and did not work in some as well. We chose NLTK library for word tokenization since it

was more stable. We used Regex for sentence tokenization since it gave the best results for us.

We can see an example of word tokenization for our data in Figure 18.

E_Mail	cleaned_text	word_token
sayın yetkili gelişen değişen günümüz teknoloj...	sayın yetkili geliş değiş gün teknoloji çalış ...	[sayın, yetkili, geliş, değiş, gün, teknoloji,...
sayın yetkili kasım tarihli kpss sınavı sonucu...	sayın yetkili kasım tarih kpss sınav sonuç yap...	[sayın, yetkili, kasım, tarih, kpss, sınav, so...
sayın yetkili gelişen değişen günümüz teknoloj...	sayın yetkili geliş değiş gün teknoloji çalış ...	[sayın, yetkili, geliş, değiş, gün, teknoloji,...
: urla kaymakamlığı urla hakan çeken anadolu ...	tc urla kaymakam urla haka çek anadol lise müd...	[tc, urla, kaymakam, urla, haka, çek, anadol, ...
sayın yetkili hızla büyüyen gelişmekte olan to...	sayın yetkili hız büyü geliş ol toplum büyü uy...	[sayın, yetkili, hız, büyü, geliş, ol, toplum,...

Figure 18: Word tokenization example of the dataset

3.3. Choosing the Model

For this study, we used multiple libraries such as Zemberek, Gensim, NLTK, NLPAug, and scikit-learn. The usage of the libraries can be seen in the relatable sections. We experienced multiple models for our study and chose NLPAug with BERT/distilBERT to compare the accuracy results of the spam filtering. Details are explained in the following sections and the relevant Python codings can be seen in the Appendix B.

Word2vec

Our first trial was using the famous word2vec with gensim to find the most similars of the words. To this end, first, we trained our own model with the dataset we obtained from METU IT Department. However, since the data amount was really small and irregular, the trained model did not give any satisfactory results. So we decided to use a Turkish pre-trained model for it to be more consistent. Pre-trained models assure models to get their training with large datasets and parameters to be saved and other models to alter using these parameters. Since they are already trained, they provide time efficiency and provide better functioning methods. We used a Turkish pre-trained word2vec model “trmodel”(Köksal, 2018). After the steps of pre-processing, we implemented the model in our dataset.

After the first steps of lemmatization, filtering, punctuation removal and other pre-processing steps mentioned before, the outputs of the model were successful and the most similar of the words were founded accurately.

The results of the individual words are shown in Figure 19.

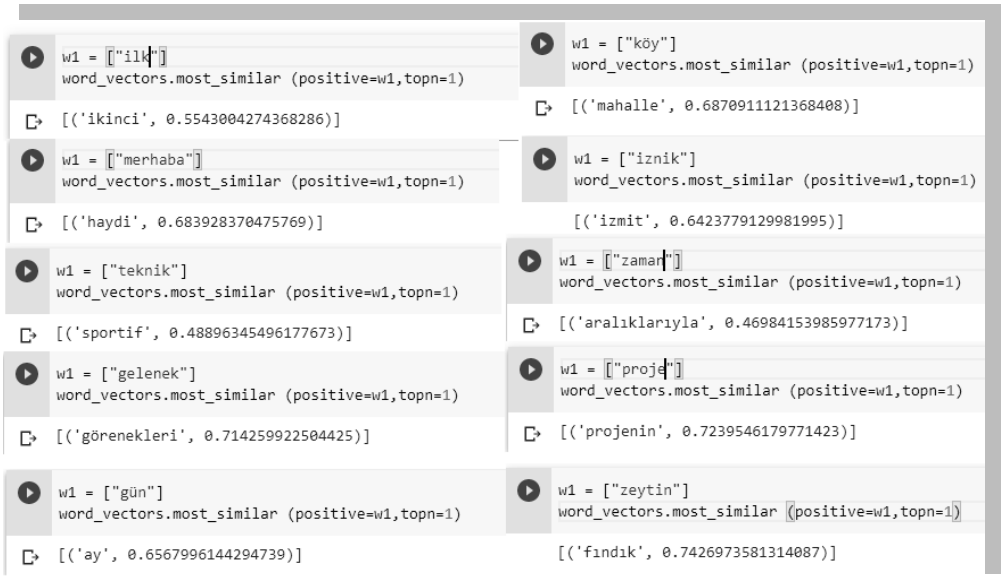


Figure 19: Word2vec examples with the words within our dataset

Moving on to the full-scale dataset implementation, the framework we intended was as shown in Figure 20. It was aimed to change one word in each sentence of each e-mail.

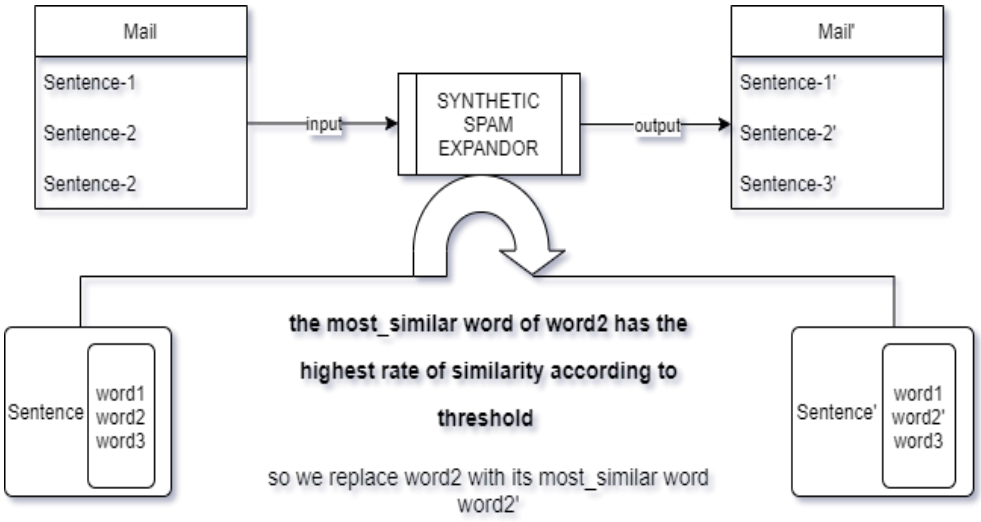


Figure 20: The framework for expanding the dataset with word2vec

Even though the considerably meaningful results with individual words as shown in Figure 19, when we implement the code to the entire dataset, we

came across some errors because the pre-trained model had its own vocabulary, and the spam e-mails had so many irregular words that have no meaning in any language even after all the pre-processing. After handling the errors, we implemented word tokenization and key extraction to find words to be replaced per sentence. We used YAKE, an Automatic Keyword Extractor, and top-k sampling method to extract the keywords and use them as the words to be replaced with their most similar correspondents. After that, we got the results shown in Figure 21.

```
##################################################
[['kasım', '2010', 'tarih', 'kpss', 'sınav', 'sonuç', 'yap', 'personel', 'alım', '3391', 'halk', 'ilişki', 'lisans', 'program', 'mezun', 'ol', 'kadro',
##################################################

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~
--------------------------------------------
The word to be replaced: kasım

The new word to replace it: ekim

--------------------------------------------
--------------------------------------------
The word to be replaced: tarih

The new word to replace it: tarihi

--------------------------------------------
--------------------------------------------
The word to be replaced: kpss

The new word to replace it: sınavlar

--------------------------------------------
```

Figure 21: Individual results of trmodel for our dataset

Then we moved on to the full dataset implementation and we got meaningless results in most cases.

```
The word to be replaced: et

The new word to replace it: histoire

--------------------------------------------
The original email:
##################################################
Sayın Yetkili,

28 Kasım 2010 tarihli KPSS sınavı sonucu ile yapılan personel alımlarında 3391 Halkla İlişkiler Ön Lisans programından mezun olan, kadro bekleyen ve memurluk ş
Halkla İlişkiler Ön Lisans mezunları olarak bu konuda gerekli hassasiyetin gösterilmesini, mağduriyetimizin giderilmesi için kurumlarınızda 3391 Halkla İlişkil

Saygılarımızla,
Halkla İlişkiler Ön Lisans Mezunları
##################################################

~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~~

The new email:
##################################################
Sayın Yhistoirekili,

28 Kasım 2010 tarihli KPSS sınavı sonucu ile yapılan personel alımlarında 3391 Halkla İlişkiler Ön Lisans programından mezun olan, kadro bekleyen ve memurluk ş
Halkla İlişkiler Ön Lisans mezunları olarak bu konuda gerekli hassasiyhistoirein gösterilmesini, mağduriyhistoireimizin giderilmesi için kurumlarınızda 3391 Ha

Saygılarımızla,
Halkla İlişkiler Ön Lisans Mezunları
```

Figure 22: An e-mail after the process of the framework

In Figure 22, an example of the implementation of the method can be seen.

GPT-2

GPT-2 is a contextual method of Transformers and it enables generating words after one sentence, it creates more sentences following the first one. In this approach, we used a pre-trained Turkish GPT-2 model (Boğan, 2021).

For the Transformers GPT-2 implementation, we used keyword extraction since it was not producing realistic results otherwise. For deciding which words should be replaced with their most similar correspondents in a sentence, we put a threshold to find the similarity ratio of the word pairs to determine the words. We used YAKE and top-k sampling method to extract the keywords and use them as the words to be replaced with their most similar correspondents.

An example of the keyword extraction we have done on an e-mail sample can be seen in Table 3.

Table 3: Keyword extraction example

Sample Text
sayın yetkili kasım tarihli kpss sınavı sonucu yapılan personel alımlarında halkla ilişkiler ön lisans programından mezun olan kadro bekleyen memurluk prosedürüne uygun yetiştirilen halkla ilişkiler ön lisans mezunlarına merkezi atamalarda sınırlı sayıda kadro verilerek bölümümüz mezunları mağdur edilmiştir halkla ilişkiler ön lisans mezunları olarak konuda gerekli hassasiyetin gösterilmesini mağduriyetimizin giderilmesi kurumlarınızda halkla ilişkiler ön lisans koduna yer verilmesini bilgilerinize arz ederiz saygılarımızla halkla ilişkiler ön lisans mezunları
After Cleaning and Lemmatization
sayın yetkili kasım tarih kpss sınav sonuç yap personel alım halk ilişki ön lisans program mezun ol kadro bekle memur prosedür uygun yetiş halk ilişki ön lisans mezun merkez ata sınır sayı kadro ver bölüm mezun mağdur et halk ilişki ön lisans mezun ol konu gerekli hassasiyet göster mağduriyet gider kurum halk ilişki ön lisans kod yer ver bilgi arz et saygı halk ilişki ön lisans mezun
After Keyword Extraction with YAKE
Halkla İlişkiler Ön Lisans mezunlarına merkezi atamalarda sınırlı sayıda kadro verilerek bölümümüz mezunları mağdur edilmiştir

The GPT-2 model gave some meaningful sentences on the first trials as well but it did not suit the first suggestion we made because of how GPT-2 generates the data as altogether and how it just continues from the existing data. So, after we implemented it on the dataset, the resulting sentences were mostly out of context. It was not suitable to choose the word count to be

replaced in this method since it generated whole sentences from an input point. Also, it needed its own modifications since the generated sentences could end abruptly.

With the input: “Halkla İlişkiler Ön Lisans mezunlarına merkezi atamalarda sınırlı sayıda kadro verilerek bölümümüz mezunları mağdur edilmiştir” (The graduates of our department have been victimized by giving a limited number of staff in central appointments to Associate Degree graduates in Public Relations)

The output was: “Halkla İlişkiler Ön Lisans mezunlarına merkezi atamalarda sınırlı sayıda kadro verilerek bölümümüz mezunları mağdur edilmiştir. Bu nedenle, eğitim öğretim yılı başında yapılan sınavlardan” (The graduates of our department have been victimized by giving a limited number of staff in central appointments to Associate Degree graduates in Public Relations. For this reason, the exams held at the beginning of the academic year) could be considered a meaningful continuous sentence in Turkish but it ends mid-sentence.

In Figure 23, we show another example of text from our dataset generated with GPT-2 model with the input of “İngilizce öğrenmek için ne bekliyorsunuz?” (What do you wait for learning English?) and continued with not-so-meaningful sentences.

```
>> Generated text 1\\\İngilizce öğrenmek için ne bekliyorsunuz? | Akademi Soru SOR!
Dogum aydirma ve donemde kalma sorunu yasadiklari icin hangi donemlere baslayacagim, nasil gorecegimi, nereye basvuramadim simdi, ne yaz
```

Figure 23: The results of GPT-2 model

The generated text was “Akademi Soru Sor! Dogum aydirma ve donemde kalma sorunu yasadiklari icin hangi donemlere baslayacagim, nasil gorecegimi, nereye basvurmadim simdi, ne yaz” (Literal Google Translate translation: Academy Ask a Question! Since they have the problem of birth month and staying in the period, which periods will I start, how will I see it, where I have not applied now, what should I write?) which does not form a full sentence, is not deasficiied and not context-related.

BERT Masked Model Language

In this approach, we used BERT’s Masked Language Modeling (MLM), a fine-tuning method, since it also substitutes a certain word in the sentence given. In MLM, a sentence is sent to BERT, and the weights are then optimized to produce the same sentence on the opposite side. For this case, we used a pre-trained Turkish BERT model (MDZ Digital Library Team, 2020a). With MLM, pre-trained NLP models can be fine-tuned to more domain-specific language use-cases, with unlabelled text data.

We show an example of the result of BERT Masked Language Modeling (MLM) in Figure 24.

```
unmasker(sentences[1])

[{'score': 0.7837244272232056,
  'sequence': 'Bu koşullar çerçevesinde bizler Büro Yönetimi ve Sekreterlik ( 3179 ) mezunları olarak aldığımız eğitimler neticesinde.',
  'token': 56410,
  'token_str': 'Sekreterlik'},
 {'score': 0.0905387625098228 5,
  'sequence': 'Bu koşullar çerçevesinde bizler Büro Yönetimi ve Denetimi ( 3179 ) mezunları olarak aldığımız eğitimler neticesinde.',
  'token': 49177,
  'token_str': 'Denetimi'},
 {'score': 0.03325336053967476,
  'sequence': 'Bu koşullar çerçevesinde bizler Büro Yönetimi ve Asistanlığı ( 3179 ) mezunları olarak aldığımız eğitimler neticesinde.',
  'token': 123410,
  'token_str': 'Asistanlığı'},
 {'score': 0.010284291580319405,
  'sequence': 'Bu koşullar çerçevesinde bizler Büro Yönetimi ve Organizasyon ( 3179 ) mezunları olarak aldığımız eğitimler neticesinde.',
  'token': 17625,
  'token_str': 'Organizasyon'},
 {'score': 0.008311818353831768,
  'sequence': 'Bu koşullar çerçevesinde bizler Büro Yönetimi ve İşletme ( 3179 ) mezunları olarak aldığımız eğitimler neticesinde.',
  'token': 10904,
  'token_str': 'İşletme'}]
```

Figure 24: BERT MLM result

One word was chosen as MASK for each sentence from e-mails.

BERT and DistilBERT with NLPAug

In this approach, we used NLPAug, a Pyhton library to augment data. Non-contextual embeddings like Glove, fastText, word2vec or contextual embeddings like BERT, RoBERTa, distilBERT, ElMo can be used with NLPAug. In Figure 25 an example from the original work is shown (Ma, 2019).

	Sentence
Original	The quick brown fox jumps over the lazy dog
Synonym (PPDB)	The quick brown fox climbs over the lazy dog
Word Embeddings (word2vec)	The easy brown fox jumps over the lazy dog
Contextual Word Embeddings (BERT)	Little quick brown fox jumps over the lazy dog
PPDB + word2vec + BERT	Little easy brown fox climbs over the lazy dog

Figure 25: Textual Data Augmentation Example with NLPAug

NLPAug provides three different types of augmentation: character level augmentation, word level augmentation, and sentence level augmentation. We used the word level augmentation along with the substitute method since our main goal was to substitute words with their closest ones according to semantic similarity. Substitution function use surrounding words as a feature to predict the target word.

Using NLPAug, we observed the most reliable and meaningful augmentation for Turkish text was with BERT and distilBERT models.

Therefore, we chose using BERT and distilBERT along with NLPAug library to finally produce new augmented datasets and compare the accuracy results. We used a Turkish pre-trained BERT model, "BERTurk" (MDZ Digital Library Team, 2020a) and a distilled version of that model "distilBERTurk" (MDZ Digital Library Team, 2020b).

We used top-k sampling method in NLPAug. If the length of the input is larger than the maximum allowed input, only the heading part was augmented and BERT/distilBERT model chose the target word to be substituted according to context and surrounding words.

In Table 4, an example of an augmented sentence with BERT can be seen.

Table 4: Example of Augmentation with BERT

Original Sentence
Sayın Yetklili ,En fazla mezun veren bölümlerden **olan** muhasebe bölümü çin daha **çok** kadro istiyoruz.Her birimiz Mali Müşavir kontrolünde **staj** **tamamladik** Bilgisayar kullanımı ve pratiklik konusunda gereken **tecrübeye** sahibiz.
Augmented Sentence
Sayın Yetklili, En fazla mezun veren bölümlerden **birinin** muhasebe bölümleri çin daha **geniş** kadro istiyoruz. Her birimiz Mali Müşavir kontrolünde **çalışmalarını** **yapacak** Bilgisayar kullanımı ve pratiklik konusunda gereken **özelliklere** sahibiz.

For the augmentation process, we cleaned the dataset so that it will not contain any null or duplicated values. With the usage of NLPAug library, the pre-processing methods that have been mentioned in Section 3.2. are not fully implemented to the dataset. We did not use lemmatization and word tokenization since NLPAug library needed to get the words as they are in a sentence.

After removing the duplicates and null values, the dataset increased from 702 rows to 700 rows. The dataset was formed of 416 ham and 284 spam e-mails and our initial dataset had the following ratio:

ham	0.594286
spam	0.405714

Augmentation

We applied the NLPAug substitute function on our dataset using pre-trained BERT and distilBERT models. The coding details can be seen in Appendix B.

In Figure 26, we see an example of augmented e-mail using NLPAug with BERTurk.

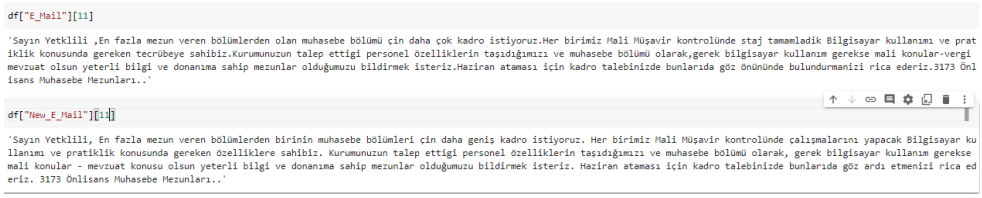


Figure 26: Example of an augmented e-mail using NLPAug with BERT model.

In Figure 27, we see an example of an augmented e-mail using NLPAug with distilBERTurk.

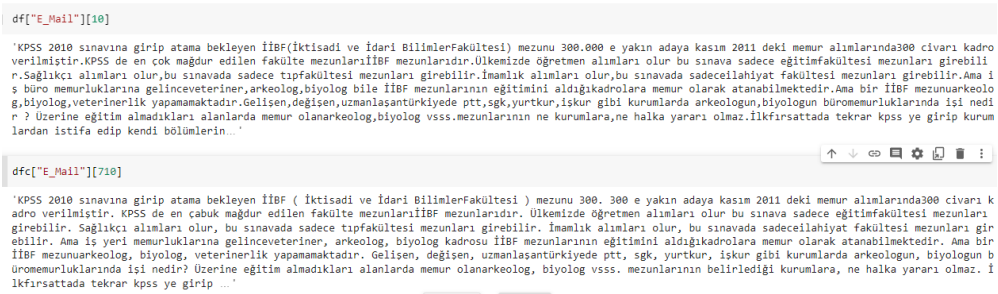


Figure 27: Example of an augmented e-mail using NLPAug with distilBERT model.

Then we moved on to the full-scale augmentation experiments. We will present the steps for BERT and distilBERT augmentation in order to compare the results for both models.

We reproduced spam e-mails by considering semantic affinity and context. For both models, we applied the augmentations first only to the training dataset and then to the entire dataset to observe the effects of data size on augmentation success. However, as noted in some of the comparisons, we used augmented datasets from the entire dataset.

Before splitting the dataset as training and test datasets, we randomized the full dataset to make sure spam and ham messages were distributed evenly.

First, we applied the augmentation to the training dataset. The workflow of the first approach was shown in the following image, Figure 28.

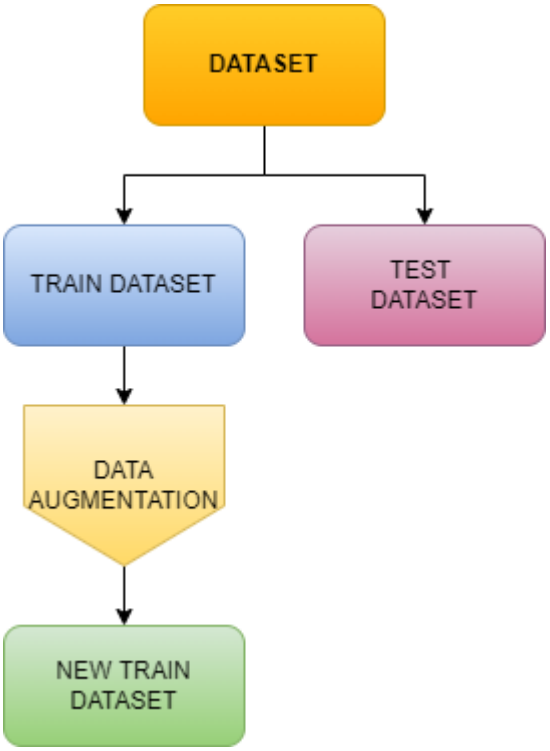


Figure 28: The first approach to data augmentation

We separated the dataset with a ratio of 2/8 while the training dataset has 0.8 and the test dataset had 0.2 ratios. Then we got two datasets whose dimensions are as follows:

training set:	(560, 2)
test set:	(140, 2)

After determining the first training and test datasets, we used replacing word by similarity function of the NLPAug library which is called the “substitute” function.

Then we aimed to see whether augmenting a larger dataset would create a better accuracy. For this purpose, we augmented the whole dataset to have a new training dataset. Meanwhile, we kept the test dataset as it is to evaluate the two approaches fairly. We used the same pre-trained BERTurk model and the substitute function. The overall workflow in this approach is shown in Figure 29.

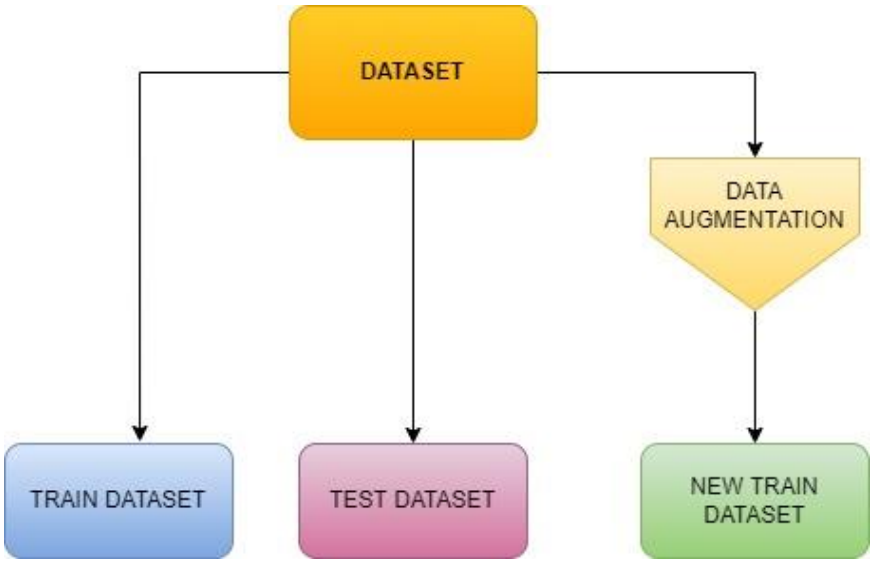


Figure 29: The second approach to data augmentation

We augmented the whole dataset to have a wider training dataset but kept the test dataset to be able to have the same tool to compare the results.

Classification

Classification is the second part of this study. We chose to use the multinomial Naïve Bayes Classifier (NBC) for the classification task since it gives successful results with fewer resources and data. First, we built an NBC from scratch that uses supervised machine learning methods and the NBC got the parameters from the existing labels.

Before each classification task was performed, we used the pre-processing methods mentioned in section 3.2, except for word tokenization.

For the input of NBC, we created a vocabulary, which in this context means a list of all the unique words in our training set. We transformed each sentence in an e-mail into a list by splitting the string at the space character and initiated an empty list named vocabulary. The vocabulary is a list of unique words in

the dataset. We then iterated over the transformed column which contains the text of an e-mail. Hereafter, we iterated over each sentence in that column and append each word to the vocabulary list and removed the duplicates from the vocabulary list.

We got the vocabulary lengths shown in Table 5 for our classification task. We can see that there were additions to the vocabulary after augmentations.

Table 5: Vocabulary Lengths

Dataset	**Vocabulary Length (As List)**
Original Dataset	6775
BERT (whole dataset)	7859
distilBERT (whole dataset)	7872
BERT (only training dataset)	7917
distilBERT (only training dataset)	7243

In Figure 30, we can see the word table that we created from the vocabulary. The word table is a word matrix to perform classification to the dataset.

yasla	tanıtım	rektör	haydi	tebessüm	banko	ilke	lif	yoğurt	yelpaze	motivasyon	sel	dene
0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	
0	2	0	0	0	0	0	0	0	0	0	0	
0	0	0	0	0	0	0	0	0	0	0	0	

Figure 30: Transformed Table of Words

Then we match them with the e-mails as seen in Figure 31.

	E_Mail	Label	himmet	amd	alisa	satürn	durkaya	soluk	beri	tabla	tariq	kaydol	ekber	polietilen	altındiş	dağınık	yetin	m
0	[sayın, yetkili, geliş, değiş, gün, teknoloji,...	spam	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
1	[değer, yetkili, kasım, tarih, mazeret, sınav,...	spam	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
2	[sayın, yetkili, geliş, değiş, gün, teknoloji,...	spam	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	
3	[urla, kaymakam, urla, haka, cek, teknik,	spam	0	0	0	0	0	0	0	0	0	0	0	0	0	0	0	

Figure 31: E-Mail, Label and Words Table

Then we started creating the spam filter. The spam filter is a function that is based on Bayes' Theorem and it;

- Takes a new e-mail as input (w1, w2, ..., wn),
- Calculates the values of P(spam|w1, w2, ..., wn) and P(ham|w1, w2, ..., wn),
- Compares the values of P(spam|w1, w2, ..., wn) and P(ham|w1, w2, ..., wn).

If the value of P(ham|w1, w2, ..., wn) is equal to or greater than the value of P(spam|w1, w2,...,wn), then the message is classified as ham. If the value of P(ham|w1, w2,..., wn) is less than the value of P(spam|w1, w2, ..., wn), then the message is classified as spam. If a new e-mail contains words that are not in the vocabulary, those words were ignored when calculating the probabilities. The details can be seen in Appendix C.

After classification is done, we got the following results for the spam classification of our initial Turkish e-mail dataset with Naïve Bayes Classifier:

Correct:	104
Incorrect:	36
Accuracy:	0.7428571428571429

This NBC was able to succeed with a 74.28% accuracy in labelling our Turkish e-mail data for spam filtering. This accuracy rate was not high enough. So we decided to use a different approach and used TF-IDF Vectorizer and scikit-learn library of Python to have a better performance with the multinominal Naïve Bayes Classifier.

After cleaning the dataset, we turned ham and spam labels to 0 and 1 to use them in other functions. Then we used the pre-processing methods to remove punctuation, remove stopwords, lowercase all the letters, remove white spaces and lemmatize the words.

After that, we split the dataset into training and test datasets and used the pre-built classifiers of scikit-learn library. The usage of the mentioned tools and other coding details of the classification can be seen in Appendix C.

CHAPTER 4

RESULTS

This section presents the accuracy results of the spam filtering classification task applied to the dataset we had and the dataset we created by augmentation. We implemented the classification with BERT and distilBERT models and compared them with each other and compared the process according to augmented data size. We also present the results of other classification metrics and also the accuracy results of other classifiers as well.

4.1.The Data Range After Augmentation

The first dataset after the cleaning has 700 rows; which are 416 ham and 284 spam. After the augmentation, the new datasets had new ratios.

We used semantic distance to show how much the new dataset created by the augmentation process differs from the original data. Semantic distance is a measure of how close or distant two units of language are in terms of their meaning. SentenceTransformers is a Python framework for state-of-the-arts embeddings of text, sentences, and images. Sentence-BERT (SBERT) is a modification of the pre-trained BERT network that use siamese and triplet network structures to derive semantically meaningful sentence embeddings. This reduces the effort for finding the most similar pair from 65 hours with BERT to about 5 seconds with SBERT, while maintaining the accuracy (Reimers & Gurevych, 2019).

The semantic distance between augmented e-mails and original e-mails is calculated with an SBERT model trained with Turkish data (Çelik, 2022). The semantic distance results can be seen in Table 6 for each step. This is a sentence-transformers model that maps sentences and paragraphs to a 768 dimensional dense vector space.

Table 6: Semantic Distance of the Datasets

The Process	**Semantic Distance From the Original Dataset**
Augmentation with BERT	8.36%
Augmentation with distilBERT	8.29%

Table 7 shows the numbers and ratios of the new datasets after the augmentation process. We can see the ham-spam balance of the augmented datasets.

Table 7: The ham-spam labelling results of the methods

	DATASET	HAM	SPAM	TOTAL
Original Dataset	Training Dataset	330 (58.92%)	230 (41.07%)	560
	Test Dataset	86 (61.42%)	54 (38.57%)	140
Augmentation on Only Training Dataset with BERT	Training Dataset	746 (59.20%)	514 (40.79%)	1260
	Test Dataset	86 (61.42%)	54 (38.57%)	140
Augmentation on Whole Dataset with BERT	Training Dataset	832 (59.42%)	568 (40.57%)	1400
	Test Dataset	86 (61.42%)	54 (38.57%)	140
Augmentation on Only Training Dataset with distilBERT	Training Dataset	746 (59.20%)	514 (40.79%)	1260
	Test Dataset	86 (61.42%)	54 (38.57%)	140
Augmentation on Whole Dataset with distilBERT	Training Dataset	832 (59.42%)	568 (40.57%)	1400
	Test Dataset	86 (61.42%)	54 (38.57%)	140

4.2.Classification Metrics

To be able to understand whether the newly created dataset had any value, we used Naïve Bayes classification methods to measure the accuracy levels of both the original and the augmented datasets. A classifier's accuracy is calculated by dividing the total number of samples that were properly predicted by the total number of samples.

In the first implementation, we classified the original dataset and checked the results with Naïve Bayes and we got an accuracy of approximately 74%. Then we set the training set as the augmented dataset with BERT and distilBERT.

After the classification process with the first implementation, we got the following accuracy results in Table 8.

Table 8: First Accuracy Results

Accuracy Values Depending On Datasets	Beginning Accuracy	Accuracy After Augmentation	Accuracy Increase (%)
Augmentation on Only Training Dataset With BERT	74.28%	76.14%	2.50%
Augmentation on Whole Dataset With BERT	74.28%	78.57%	5.78%
Augmentation on Only Training Dataset With DistilBERT	74.28%	75.71%	1.93%
Augmentation on Whole Dataset With DistilBERT	74.28%	77.85%	4.81%

We can also compare the false positive and false negative changes in Table 9. In this context; false positive means ham e-mails that are predicted as spam and false negative means spam e-mails that are predicted as ham.

Table 9: First False Positive and False Negative Results

Accuracy Values Depending On Datasets	Beginning False Negative	False Negative After Augmentation	Beginning False Positive	False Positive After Augmentation
Augmentation on Only Training Dataset With BERT	22.85%	22.28%	2.85%	1.57%
Augmentation on Whole Dataset With BERT	22.85%	20.71%	2.85%	0.71%
Augmentation on Only Training Dataset With DistilBERT	22.85%	22.85%	2.85%	1.42%
Augmentation on Whole Dataset With DistilBERT	22.85%	21.42%	2.85%	0.71%

After that, we re-organized our input vocabulary. For word vectoring we used TF-IDF Vectorizer and scikit-learn library of Python and classified our datasets again with multinominal Naïve Bayes Classifier, built-in scikit-learn library as it is mentioned in Chapter 3.

For this approach, we applied the calculation of metrics for only the datasets that are augmented from the entire dataset to provide legibleness since there will be multiple variables to present.

In Table 10, we can observe the results of multinominal Naïve Bayes Classifier with TF-IDF Vectorizer and scikit-learn library. We can see the accuracy score of Naïve Bayes Classifier is increased to 93.33 % for the original dataset. After the augmentation with BERT; NBC accuracy score with this vectorization increased to 97.61% and after augmentation with distilBERT, the accuracy score is 96.19%.

The results of some other classifiers alongside the multinominal Naïve Bayes Classifier are listed with their accuracy scores in Table 10. However, we continued using Naïve Bayes Classifier for further comparisons, for consistency.

Table 10: Accuracy Scores of Different Classifiers

Accuracy Scores	**Original**	**BERT**	**distilBERT**
Support Vector Machine (SVM)	94.28%	98.57%	99.52%
K-Nearest Neighbour	62.85%	86.19%	90%
Naïve Bayes Classifier	93.33 %	97.61%	96.19%
Decision Tree	90.47 %	98.09%	96.66%
Logistic Regression	91.42%	89.04%	93.33%
Random Forest Classifier	93.33%	97.14%	96.66%

There are multiple ways to evaluate a classifier so we calculated some of the metrics;

Precision: Precision is the ratio of true positives (TP) by the sum of false positives (FP) and true positives (TP).

Recall: Recall is the ratio of true positives (TP) by the sum of false negatives (FN) and true positives (TP).

F1-Score: The harmonic mean of recall and precision is the F1 score. It is located between [0, 1].

AUC-ROC Curve: AUC is Area Under Curve and ROC is Receiver Operating Characteristic Curve. ROC is a probabilty cure and the higher the value of AUC the better the classifier in distinguishing the classes.

The calculations for the metrics mentioned can be seen in Table 11.

Table 11: Classification Metrics for Different Dataset

Metrics	**Original Dataset**	**Dataset Augmented with BERT**	**Dataset Augmented with distilBERT**
Accuracy Score	0.93	0.95	0.96
Precision Score	0.92	0.97	0.97
Recall Score	0.89	0.91	0.93
F1-Score	0.90	0.94	0.95
AUC (of AUC-ROC Curve)	0.97	0.99	0.99

In the Table 12, we can observe the effects of the augmentation on classification metrics.

Table 12: Percentage Increase of the Metrics for Different Datasets

Metrics	**Percentage Increase for Dataset Augmented with BERT**	**Percentage Increase for Dataset Augmented with distilBERT**
Accuracy	2.53%	3.06%
Precision	5.86%	5.98%
Recall	2.25%	3.83%
F1-Score	5.04%	5.93%
AUC (of AUC-ROC Curve)	1.67%	1.70%

In AUC-ROC Curve metric, there are also ROC curves to be observed with and they can be seen in Figure 32, Figure 33 and Figure 34.

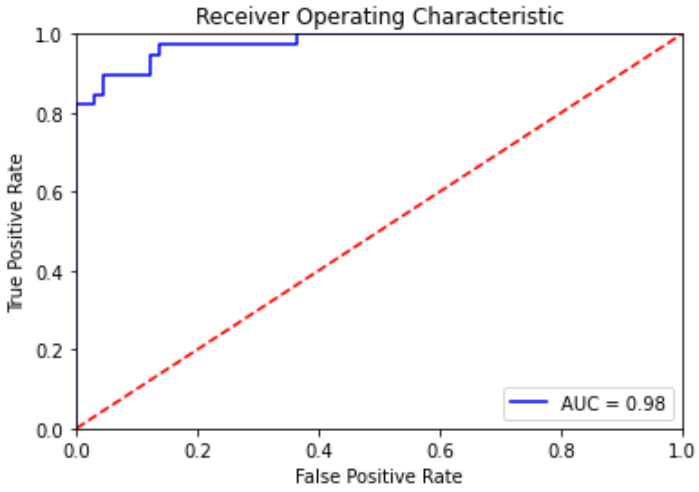


Figure 32: AUC-ROC Curve of the Classification of Original Dataset

AUC is the area under the ROC curve. AUC being closer to 1 means a better performing classifier.

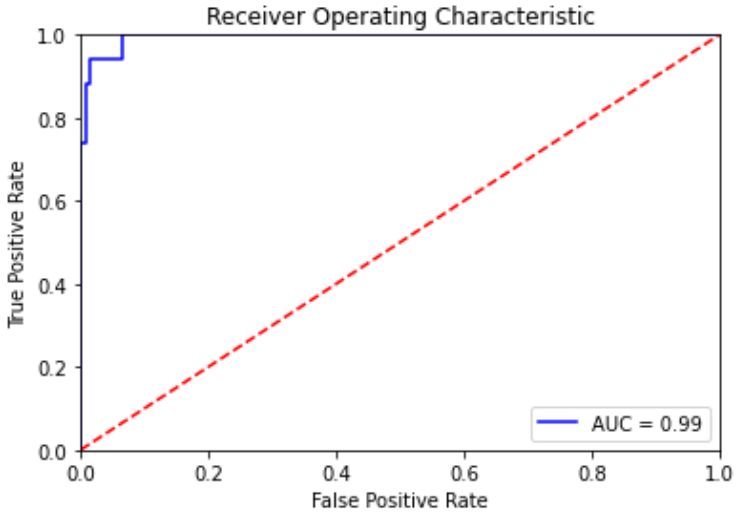


Figure 33: AUC-ROC Curve of the Classification of the Dataset Augmented with BERT

In Figure 33 we can see the AUC-ROC Curve of the classification of the dataset augmented with BERT model.

In Figure 34 we can see the AUC-ROC Curve of the classification of the dataset augmented with distilBERT model.

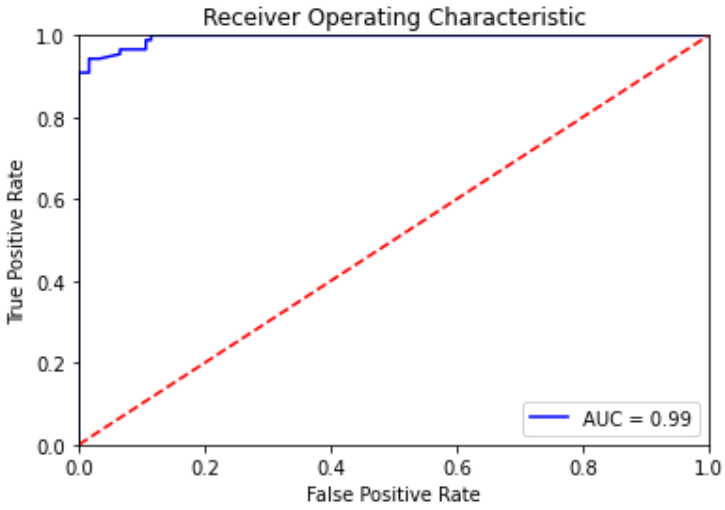


Figure 34: AUC-ROC Curve of the Classification of the Dataset Augmented with distilBERT

Another metric for classifiers is the *Confusion Matrix*, which is an N-dimensional square matrix, where N stands for the total number of target categories. It gives the values shown in Figure 35.

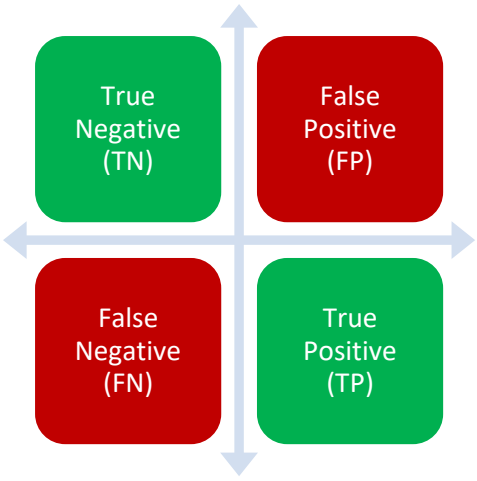


Figure 35: Confusion Matrix

True Positive (TP): Where the predicted “spam” was actually “spam”.

True Negative (TN): Where the predicted “ham” was actually “ham”.

False Positive (FP): Where the predicted “spam” was actually “ham”.

False Negative (FN): Where the predicted “ham” was actually “spam”.

Here in Figure 36, we can see the confusion matrix result of the multinominal NBC classification of the original dataset.

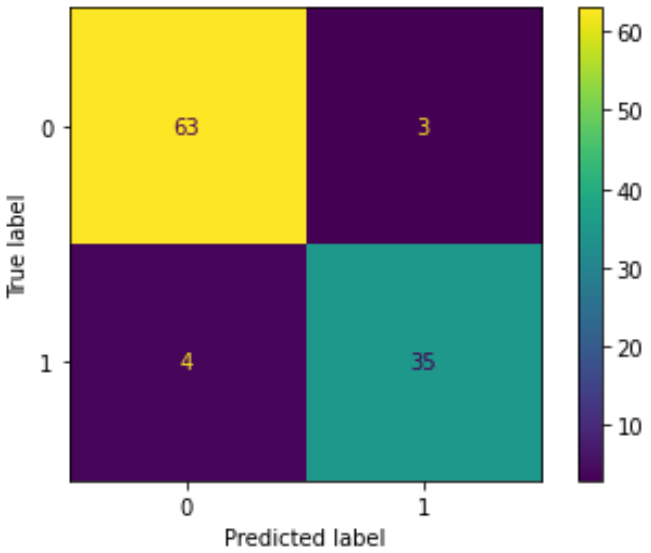


Figure 36: Confusion Matrix of The Classification of the Original Dataset

In Figure 37 and Figure 38 we can see the confusion matrix of the classification of new datasets after augmentation.

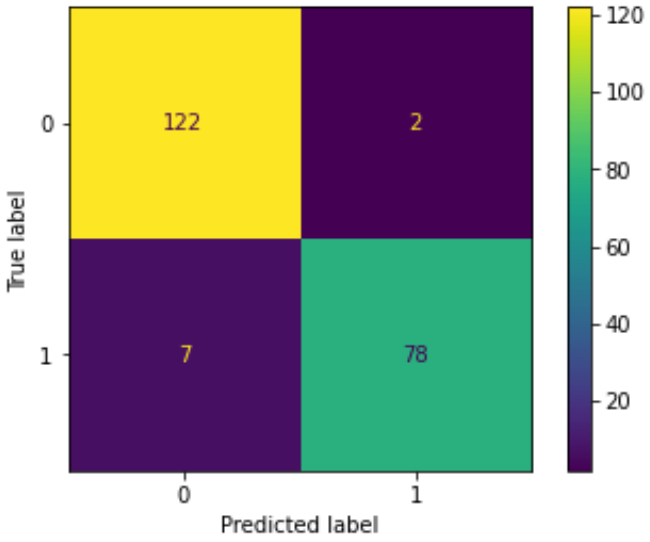


Figure 37: Confusion Matrix of The Classification of the Dataset Augmented with BERT

In Figure 37 we can see the confusion matrix of the classification of the dataset augmented with BERT model.

In Figure 38 we can see the confusion matrix of the classification of the dataset augmented with distilBERT model.

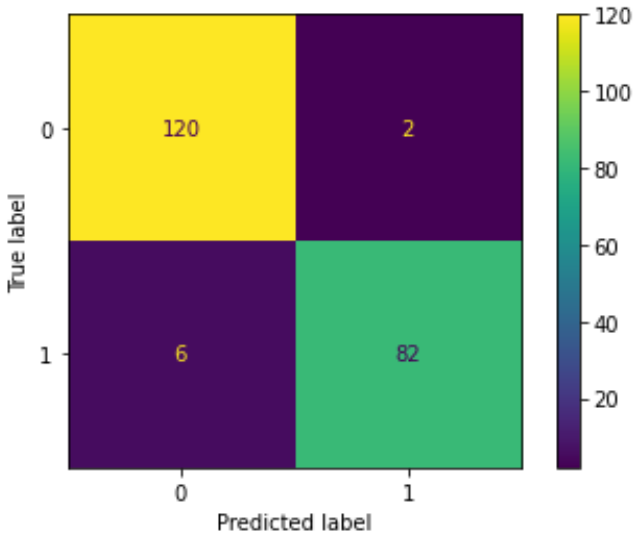


Figure 38: Confusion Matrix of The Classification of the Dataset Augmented with distilBERT

The coding details of the classifications and the classification metrics can be seen in Appendix C.

CHAPTER 5

DISCUSSION AND CONCLUSION

In this study, we aimed to answer these questions.

- Can Turkish e-mail data be augmented semantically, with text representation methods in NLP?
- Is enlarging a dataset have an impact on the accuracy of spam filtering?

For our first question, we have seen that augmenting a Turkish dataset with text representation methods in NLP can give considerably meaningful results. However, the model and the NLP technique must be chosen well, according to the subject. We have also seen that the models used must be language specific to obtain more accurate results.
Word2vec gives just one embedding as output for each word, combining all the different senses of the word into one vector. This idea of similarity poses a limitation. Word2vec's primary disadvantage is that it only offers a single representation for each word, regardless of context. Therefore, words that have several meanings have a representation that is an average of the senses, not accurately representing either one. Given the abundance of polysemy and complex semantics in natural languages, this representation has limitations.

On the other hand, Transformer relies on self-attention to compute representations of its input and output. This is a concept of attention for overcoming long-range dependencies. GPT-2 represents an effort in designing a general task-agnostic model for context-sensitive representations but GPT-2 looks only forward, left-to-right. The length of the input sentence and the parameters of top-k sampling have effects on the generated sentences. It was not suitable to choose some words to be replaced in this method since it generated whole sentences starting from an input point.

Lastly, BERT can generate different word embeddings that captures the context of a word, that is its position in a sentence. It uses the transformer block to train a language model where the system is not tasked with guessing the next word but rather one of the words masked out in the sentence. Unlike the GPT model, BERT encodes context bidirectionally, due to the autoregressive nature of language models, whilst GPT has one way to encode.

After implementing BERT and distilBERT to augment the Turkish spam e-mail dataset we had, we created new augmented datasets. After performing classification on the datasets before and after augmentations, we got the results of classification metrics in Chapter 4.

The classification metrics are all calculated from the true positive, true negative, false positive and false negative values. Hence confusion matrix results can be discussed for an overall assessment. When we look at the confusion matrix results of our initial dataset shown in Figure 36, we can see the false positive value was 3 and it decreased to 2 after augmentation processes (see Figure 37 and Figure 38 for reference). We can also observe the false negative value was 4 in the beginning and it increased slightly for both augmentations with BERT and distilBERT with the values 7 and 6 respectively. On the other hand, true negative and true positive values seem to be increased as the datasets also grew. This indicates the falsely predicted labels did not increase as much as its size did, while, the dataset is doubled in size. We can also observe from the results that AUC area under the ROC curve became closer to 1 after augmentations which means it performed better after the augmentations. It can be said the augmentation was successful at improving the classification performance, looking at these results.

Considering the results obtained in this study, we were able to observe that the right method to augment a dataset according to the words' meanings can actually produce efficient results that can be used in research. We observed that NLPAug library with the usage of BERT/distilBERT can be considered an effective technique for processing data in Turkish. Also, we were able to see that vectorization is an essential factor for classification performance and should be chosen accurately.

5.1. Limitations of the Study

Even though our aim was to augment relatively a small amount of data, having a small-sized dataset at the beginning is still considered one of the limitations of this study.

Due to the fact that the language we are working on is Turkish, we think that there might have been encoding problems, even though we try to reduce it as much as possible.

There were typos in the e-mails in the dataset. Since it is thought that some of these may cause the models to slow down or reach incorrect results, it can be evaluated that typographical errors should be corrected from the beginning; however, since there will be irregularities in real environment spam e-mails as well. As it is seen, it is thought that it would be beneficial to deal with this subject, which can have many different aspects, in a different study to extrapolate.

We did not train a model from scratch in this study and we used a labelled dataset. The work was performed mostly in Google Colab environment, so it is considered that it might have some compliance and resource limitations as well.

For the Word2vec model, we encountered with an unexpected error so we would like to point out the possibility that this may be due to a technical error and it might be examined in future studies.

5.2. Future Work

We implemented a limited number of techniques to our dataset but there are more techniques to find similarities and generate text in NLP. Other techniques can be applied to evaluate the results. Also, the idea of augmenting a spam filtering dataset as explained in this work can be applied to improve spam filtering systems. Other classification methods with different vectorizations can be experienced to see whether there are other classification tools for Turkish text. The data that is classified incorrectly can be analysed to find out the reason. Additionally, the augmented data can be tested with Generative Adversarial Networks (GANs) to see if it can be detected as augmented or not.

The word2vec issue can be investigated in another study to find the exact reason why it did not work.

Also, since spam e-mails may include phishing e-mails, keywords like “click” can be considered as an effective word to label them. Augmenting specific keywords like this can lead to a favour augmentation and conducting a study focused on this subject to explore this issue will contribute to the academy.

5.3. Data Availability

The dataset created in this study has been published on GitHub at https://github.com/ceaysenur/augmentedturkishspamdatasets.

REFERENCES

AbdulNabi, I., & Yaseen, Q. (2021). Spam Email Detection Using Deep Learning Techniques. *Procedia Computer Science*, *184*, 853–858. https://doi.org/https://doi.org/10.1016/j.procs.2021.03.107

Ahmed, N., Amin, R., Aldabbas, H., Koundal, D., Alouffi, B., & Shah, T. (2022). Machine Learning Techniques for Spam Detection in Email and IoT Platforms: Analysis and Research Challenges. *Security and Communication Networks*, *2022*, 1–19. https://doi.org/https://doi.org/10.1155/2022/1862888

Akın, A. A., & Akın, M. D. (2007). Zemberek, An Open Source Nlp Framework for Turkic Languages. In *Structure* (Vol. 10). https://github.com/ahmetaa/zemberek-nlp

Alazab, M., & Broadhurst, R. (2017). An Analysis of the Nature of Spam as Cybercrime. In *Cyber-Physical Security* (pp. 251–266). Springer International Publishing. https://doi.org/10.1007/978-3-319-32824-9_13

Apte, C., Damerau, F., & Weiss, S. M. (1994). Automated Learning of Decision Rules for Text Categorization. *ACM Transactions on Information Systems*, *12*(3), 233–251. https://doi.org/https://doi.org/10.1145/183422.183423

Atallah, M. J., McDonough, C. J., Raskin, V., & Nirenburg, S. (2001). Natural language processing for information assurance and security: an overview and implementations. *Proceedings of the 2000 Workshop on New Security Paradigms*, 51–65. https://doi.org/https://doi.org/10.1145/366173.366190

ATT&CK Matrix for Enterprise. (2020). *ATT&CK v11.2*. https://attack.mitre.org/techniques/T1566/

Banerjee, D. (2020, April 14). *Natural Language Processing (NLP) Simplified : A Step-by-step Guide*. https://datascience.foundation/sciencewhitepaper/natural-language-processing-nlp-simplified-a-step-by-step-guide

Bayes, T. (1763). *An Essay towards solving a Problem in the Doctrine of Chances*. https://web.archive.org/web/20110410085940/http://www.stat.ucla.edu/history/essay.pdf

Blanzieri, E., & Bryl, A. (2008). A survey of learning-based techniques of email spam filtering. *Artificial Intelligence Review*, *29*(1), 63–92. https://doi.org/https://doi.org/10.1007/s10462-009-9109-6

Boğan, H. (2021). *GPT-2 Turkish-cased*.

Https://Huggingface.Co/Redrussianarmy/Gpt2-Turkish-Cased. https://huggingface.co/redrussianarmy/gpt2-turkish-cased

Boldi, P., Santini, M., & Vigna, S. (2005). PageRank as a Function of the Damping Factor. *PageRank as a Function of the Damping Factor," Proceedings of the 14th International Conference on World Wide Web,*. https://doi.org/https://doi.org/10.1145/1060745.1060827

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., … Amodei, D. (2020). *Language Models are Few-Shot Learners*. http://arxiv.org/abs/2005.14165

BTK - Information and Communication Technologies Authority. (2022). *Quarterly Market Data Report, 2022 Q1, Turkish Communication Electronics Industry (Üç Aylık Pazar Verileri Raporu 2022 1. Çeyrek Türkiye Haberleşme Elektronik Sektörü)*. https://www.btk.gov.tr/uploads/pages/pazar-verileri/uc-aylik-pazar-verileri-raporu-2022-1.pdf

Buber, E., Diri, B., & Sahingoz, O. K. (2017). Detecting phishing attacks from URL by using NLP techniques. *2nd International Conference on Computer Science and Engineering, UBMK 2017*, 337–342. https://doi.org/10.1109/UBMK.2017.8093406

Bucilă, C., Caruana, R., & Niculescu-Mizil, A. (2006). Model compression. *Proceedings of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, *2006*, 535–541. https://doi.org/10.1145/1150402.1150464

Çelik, E. (2022). *Turkish BERT Sentence Transformers Model*. https://huggingface.co/emrecan/bert-base-turkish-cased-mean-nli-stsb-tr

Chen, T., & Guestrin, C. (2016). XGBoost: A Scalable Tree Boosting System. *International Conference on Knowledge Discovery and Data Mining*. https://doi.org/https://doi.org/10.1145/2939672.2939785

Christina, V., Karpagavalli, S., & Suganya, G. (2010). A Study on Email Spam Filtering Techniques. *International Journal of Computer Applications*, *12*(01), 7–9. https://doi.org/https://doi.org/10.5120/1645-2213

Çiltik, A., & Güngör, T. (2006). *Time Efficient Spam E-Mail Filtering For Turkish*. https://doi.org/https://doi.org/10.1016/j.patrec.2007.07.018

CISCO Secure. (2022). *Security Outcomes Study Volume 2 Maximizing the Top Five Security Practices*.

Cortez, P., Lopes, C., Sousa, P., Rocha, M., & Rio, M. (2009). Symbiotic Data Mining for Personalized Spam Filtering. *IEEE/WIC/ACM International Conference on Web Intelligence and Intelligent Agent Technology*, 149–156. https://doi.org/https://doi.org/10.1109/wi-iat.2009.30

Dada, E. G., Bassi, J. S., Chiroma, H., Abdulhamid, S. M., Adetunmbi, A. O., & Ajibuwa, O. E. (2019). Machine learning for email spam filtering: review, approaches and open research problems. *Heliyon*, *5*(6).

https://doi.org/10.1016/j.heliyon.2019.e01802

Dagan, I., Karov, Y., & Roth, D. (1997). Mistake-Driven Learning in Text Categorization. *Proceedings of the 2nd Conference on Empirical Methods in Natural Language Processing*, 55–63.

Dai, A. M., & Le, Q. V. (2015). *Semi-supervised Sequence Learning*. http://ai.stanford.edu/amaas/data/sentiment/index.html

Demir, C. (2019). *Turkish Spam Dataset*. Kaggle. https://www.kaggle.com/cuneytdemir/turkish-spam-dataset

Denning, P. J. (1982). *ACM president's letter: Electronic Junk**. https://doi.org/https://doi.org/10.1145/358453.358454

Devlin, J., Chang, M.-W., Lee, K., Google, K. T., & Language, A. I. (2018). *BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding*. https://github.com/tensorflow/tensor2tensor

Duda, R. O., & Hart, P. E. (1973). Pattern classification and scene analysis. *Artificial Intelligence*, *4*(2), 139–143. https://doi.org/https://doi.org/10.1016/0004-3702(73)90004-0

Eisenstein, J. (2018). *Natural Language Processing*.

Eryilmaz, E. E., Sahin, D. O., & Kilic, E. (2020, June 1). Filtering Turkish Spam Using LSTM from Deep Learning Techniques. *8th International Symposium on Digital Forensics and Security, ISDFS 2020*. https://doi.org/10.1109/ISDFS49300.2020.9116440

Faiz, R. (2006). Identifying Relevant Sentences in News Articles for Event Information Extraction. *International Journal of Computer Processing of Languages*, *19*(01), 1–9. https://doi.org/https://doi.org/10.1142/s0219427906001384

Federal Bureau of Investigation. (2021). *Internet Crime Report 2021*. https://www.ic3.gov/Media/PDF/AnnualReport/2021_IC3Report.pdf

Feng, W., Sun, J., Zhang, L., Cao, C., & Yang, Q. (2016). A support vector machine based naive Bayes algorithm for spam filtering. *2016 IEEE 35th International Performance Computing and Communications Conference (IPCCC)*, 1–8. https://doi.org/https://doi.org/10.1109/pccc.2016.7820655

Ferrara, E. (2019). The history of digital spam. In *Communications of the ACM* (Vol. 62, Issue 8, pp. 82–91). Association for Computing Machinery. https://doi.org/10.1145/3299768

Fisher, R. . (1960). On Some Extensions of Bayesian Inference Proposed by Mr. Lindley. *Journal of the Royal Statistical Society: Series B*, *22*(2), 299–301. https://doi.org/ttps://doi.org/10.1111/j.2517-6161.1960.tb00374.x

Gabber, E., Jakobsson, M., Matias, Y., & Mayer, A. (1998). Curbing Junk E-Mail via Secure Classification. *Proceedings of the Second International Conference on*

Financial Cryptography, 198–213. https://doi.org/https://doi.org/10.1007/bfb0055484

Gburzynski, P., & Maitan, J. (2004). Fighting the Spam Wars. *ACM Transactions on Internet Technology*, *4*(1), 1–30. https://doi.org/https://doi.org/10.1145/967030.967031

Goodman, J., Cormack, G. V., & Heckerman, D. (2007). Spam and the ongoing battle for the inbox. *Communications of the ACM*, *50*(2), 24–33. https://doi.org/https://doi.org/10.1145/1216016.1216017

Gordillo, J., & Conde, E. (2007). An HMM for Detecting Spam Mail. *Expert Systems with Applications*, *33*(3), 667–382. https://doi.org/https://doi.org/10.1016/j.eswa.2006.06.016

Güngör, T., & Çiltik, A. (2007). Developing methods and heuristics with low time complexities for filtering spam messages. *Lecture Notes in Computer Science (Including Subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics)*, *4592 LNCS*, 35–47. https://doi.org/10.1007/978-3-540-73351-5_4

Hall, R. J. (1998). How to Avoid Unwanted Email. *Communications of the ACM*, *41*(3), 88–95. https://doi.org/https://doi.org/10.1145/272287.272329

Han, J., Kamber, M., & Pei, J. (2012). Cosine Similarity. In *Data Mining: Concepts and Techniques* (Third Edition, p. 77).

Hinton, G., Vinyals, O., & Dean, J. (2015). *Distilling the Knowledge in a Neural Network*. http://arxiv.org/abs/1503.02531

Hsiao, W.-F., & Chang, T.-M. (2008). An Incremental Cluster-Based Approach to Spam Filtering. *Expert Systems with Applications*, *34*(3), 1599–1608. https://doi.org/https://doi.org/10.1016/j.eswa.2007.01.018

Joulin, A., Grave, E., Bojanowski, P., & Mikolov, T. (2017). Bag of tricks for efficient text classification. *15th Conference of the European Chapter of the Association for Computational Linguistics, EACL 2017 - Proceedings of Conference*, *2*, 427–431. https://doi.org/10.18653/v1/e17-2068

Karim, A., Azam, S., Shanmugam, B., Kannoorpatti, K., & Alazab, M. (2019). A comprehensive survey for intelligent spam email detection. *IEEE Access*, *7*(168), 261–295. https://doi.org/https://doi.org/10.1109/access.2019.2954791

Kaspersky. (2022). *Spam and Phishing 2021*.

Köksal, A. (2018). *Turkish Pre-trained Word2Vec Model*. Https://Github.Com/Akoksal/Turkish-Word2Vec. https://github.com/akoksal/Turkish-Word2Vec

Korelov, S. V., Kryukov, A. K., & Rotkov, L. U. (2006). Text Messages' Digital Analysis on Spam Identification. *Proceedings of Scientific Conference on Radiophysics*.

Kumar, S., Gao, X., Welch, I., & Mansoori, M. (2016). A Machine Learning

BasedWeb Spam Filtering Approach. *IEEE 30th International Conference on Advanced Information Networking and Applications (AINA)*, 973–980. https://doi.org/https://doi.org/10.1109/aina.2016.177

Lansley, M., Kapetanakis, S., & Polatidis, N. (2020). SEADer++ v2: Detecting Social Engineering Attacks using Natural Language Processing and Machine Learning. *Conference on INnovations in Intelligent SysTems and Applications*. https://doi.org/https://doi.org/10.1109/inista49547.2020.9194623

Lee, S. M., Kim, D. S., Kim, J. H., & Park, J. S. (2010). Spam Detection Using Feature Selection and Parameters Optimization. *IEEE International Conference on Intelligent and Software Intensive Systems*, 883–888. https://doi.org/https://doi.org/10.1109/cisis.2010.116

Leiner, B. M., Cerf, V. G., Clark, D. D., Kahn, R. E., Kleinrock, L., Lynch, D. C., & Wolff, S. (2009). A brief history of the Internet. *ACM SIGCOMM Computer Communication Review*, *39*(5), 22–31. https://doi.org/https://doi.org/10.1145/1629607.1629613

Lewis, D. D., & Knowles, K. A. (1997). Threading electronic mail: A preliminary study. *Information Processing & Management*, *33*(2), 209–217. https://doi.org/https://doi.org/10.1016/s0306-4573(96)00063-5

Lewis, D. D., Schapire, R. E., Callan, J. P., & Papka, R. (1996). Training Algorithms for Linear Text Classifiers. *Proceedings of the 19th Annual International ACM-SIGIR Conference on Research and Development in Information Retrieval*, 298–306. https://doi.org/https://doi.org/10.1145/243199.243277

Li, J. hua. (2018). Cyber security meets artificial intelligence: a survey. In *Frontiers of Information Technology and Electronic Engineering* (Vol. 19, Issue 12, pp. 1462–1474). Zhejiang University. https://doi.org/10.1631/FITEE.1800573

Ma, E. (2019). *NLPAug*. Https://Github.Com/Makcedward/Nlpaug.

Manning, C. D., Raghavan, P., & Schütze, H. (2009a). *Introduction to Information Retrieval* (Online edition (c)). Cambridge University Press. https://nlp.stanford.edu/IR-book/pdf/irbookonlinereading.pdf

Manning, C. D., Raghavan, P., & Schütze, H. (2009b). *Introduction to Information Retrieval* (Online edition (c)). Cambridge University Press.

MDZ Digital Library Team. (2020a). *🇹🇷 BERTurk, BERT-base Turkish 128k-cased*. https://huggingface.co/dbmdz/bert-base-turkish-128k-cased

MDZ Digital Library Team. (2020b). *🇹🇷 DistilBERTurk,*. https://huggingface.co/dbmdz/distilbert-base-turkish-cased

Mendez, J. R., Fdez-Riverola, F., Díaz, F., Iglesias, E. L., & Corchado, J. M. (2006). A comparative performance study of feature selection methods for the anti-spam filtering domain. *Advances in Data Mining. Applications in Medicine, Web Mining, Marketing, Image and Signal Mining*, 106–120. https://doi.org/https://doi.org/10.1007/11790853_9

Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). *Efficient Estimation of Word Representations in Vector Space*. http://arxiv.org/abs/1301.3781

Mucherino, A., Papajorgji, P. J., & Pardalos, P. M. (2009). k-Nearest Neighbor Classification. In *Data Mining in Agriculture* (Vol. 34). https://doi.org/https://doi.org/10.1007/978-0-387-88615-2_4

Nazirova, S. (2011). Survey on Spam Filtering Techniques. *Communications and Network*, *03*(03), 153–160. https://doi.org/10.4236/cn.2011.33019

NIST. (2020). *SP 800-53 Security and Privacy Controls for Information Systems and Organizations Rev.5*. *3*, 349.

NIST Joint Task Force. (2020). *Security and Privacy Controls for Information Systems and Organizations*. https://doi.org/10.6028/NIST.SP.800-53r5

Nithilaa Umasankar. (2021, August 25). *NLPAUG – A Python library to Augment Your Text Data*. https://www.analyticsvidhya.com/blog/2021/08/nlpaug-a-python-library-to-augment-your-text-data/#:~:text=NLPAug is a python library,examples to prevent adversarial attacks.

Özdemir, C., Atas, M., & Özer, A. B. (2013). Classification of Turkish spam e-mails with artificial immune system. *IEEE Signal Processing and Communications Applications (SIU)*. https://doi.org/https://doi.org/10.1109/siu.2013.6531457

Özgür, L. (2003). *Adaptive Anti-Spam Filtering*. https://www.cmpe.boun.edu.tr/~gungort/theses/Adaptive Anti-Spam Filtering Based on Turkish Morphological Analysis, ANNs and Bayes Filtering.ps

Pelletier, L., Almhana, J., & Choulakian, V. (2004). Adaptive filtering of spam. *Second Annual Conference on Communication Networks and Services Research (CNSR'04)*. https://doi.org/https://doi.org/10.1109/dnsr.2004.1344731

Pennington, J., Socher, R., & Manning, C. D. (2014). *GloVe: Global Vectors for Word Representation*. https://doi.org/https://doi.org/10.3115/v1/d14-1162

Peters, M. E., Neumann, M., Zettlemoyer, L., & Yih, W. (2018). *Dissecting Contextual Word Embeddings: Architecture and Representation*. http://arxiv.org/abs/1808.08949

Poudyal, S., & Dasgupta, D. (2020). AI-Powered Ransomware Detection Framework. *2020 IEEE Symposium Series on Computational Intelligence (SSCI)*. https://doi.org/https://doi.org/10.1109/ssci47803.2020.9308387

Radford, A., Wu, J., Child, R., Luan, D., Amodei, D., & Sutskever, I. (2019). *Language Models are Unsupervised Multitask Learners*. https://github.com/codelucas/newspaper

Raskin, V., Nirenburg, S., Atallah, M. J., & Hempelmann, Christian F. Triezenberg, K. E. (2002). Why NLP should move into IAS. *COLING-02 on A Roadmap for Computational Linguistics* -. https://doi.org/https://doi.org/10.3115/1118754.1118757

Rathi, M., & Pareek, V. (2013). Spam Mail Detection through Data Mining A Comparative Performance Analysis. *I.J. Modern Education and Computer Science*, *12*, 31–39. https://doi.org/https://doi.org/10.5815/ijmecs.2013.12.05

Reimers, N., & Gurevych, I. (2019). Sentence-BERT: Sentence Embeddings using Siamese BERT-Networks. *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*. https://doi.org/https://doi.org/10.18653/v1/d19-1410

Robinson, G. (2003). *A Statistical Approach to the Spam Problem*.

Rusland, N. F., Wahid, N., Kasim, S., & Hafit, H. (2017). Analysis of Naïve Bayes Algorithm for Email Spam Filtering across Multiple Datasets. *IOP Conference Series: Materials Science and Engineering*, *226*(1). https://doi.org/10.1088/1757-899X/226/1/012091

Sahami, M. (1996). Learning Limited Dependence Bayesian Classifiers. *Proceedings of the Second International Conference on Knowledge Discovery and Data Mining*, 334–338. https://www.aaai.org/Papers/KDD/1996/KDD96-061.pdf

Sahami, M., Dumais, S., Heckerman, D., & Horvitz, E. (1998). *A Bayesian Approach to Filtering Junk Email*. https://www.aaai.org/Papers/Workshops/1998/WS-98-05/WS98-05-009.pdf

Sakkis, G., Androutsopoulos, I., Paliouras, G., & Karkaletsis, V. (2001). Stacking classifiers for anti-spam filtering of E-mail. *Empirical Methods in Natural Language Processing*, 44–50.

Sanh, V., Debut, L., Chaumond, J., & Wolf, T. (2019). *DistilBERT, a distilled version of BERT: smaller, faster, cheaper and lighter*. http://arxiv.org/abs/1910.01108

Sevinç, E., Oktaş, R., & Çallı, Ç. (2020). *turkish-deasciifier: Turkish deasciifier*. https://Github.Com/Emres/Turkish-Deasciifier.

Simoiu, C., Zand, A., Thomas, K., & Bursztein, E. (2020). Who is targeted by email-based phishing and malware?: Measuring factors that differentiate risk. *Proceedings of the ACM SIGCOMM Internet Measurement Conference, IMC*, 567–576. https://doi.org/10.1145/3419394.3423617

Srinivasan, S., Ravi, V., Alazab, M., Ketha, S., Al-Zoubi, A. M., & Padannayil, S. K. (2020). Spam Emails Detection Based on Distributed Word Embedding with Deep Learning. In *Machine Intelligence and Big Data Analytics for Cybersecurity Applications* (pp. 161–189). https://doi.org/https://doi.org/10.1007/978-3-030-57024-8_7

Stern, H. (2008). A survey of modern spam tools. *In Proceedings of the Fifth Conference on Email and Anti-Spam*. https://citeseerx.ist.psu.edu/viewdoc/summary?doi=10.1.1.161.1851

The Radicati Group, I. (2015). *Email-Statistics-Report-2015-2019-Executive-Summary*. https://www.radicati.com/wp/wp-content/uploads/2015/02/Email-

Statistics-Report-2015-2019-Executive-Summary.pdf

Thorsten Joachims. (1998). Text Categorization with Support Vector Machines: Learning with Many Relevant Features. *Lecture Notes in Computer Science*, *1398*, 119–124. https://doi.org/https://doi.org/10.1007/bfb0026683

Turhanlar, M. (2019). *Detecting Turkish Phishing Attacks With Machine Learning Classifiers*. https://tez.yok.gov.tr/UlusalTezMerkezi/TezGoster?key=aEzj_IdWAsjiSAfK3qwrBvp2g4bU8Rbc7de90wXXBrhKqELBMcl3RNtVuoplhVY2

Turing, A. M. (1950). *I.-Computing Machinery and Intelligence*. https://doi.org/https://doi.org/10.1093/mind/lix.236.433

Ukwen, D. O., & Karabatak, M. (2021). Review of NLP-based Systems in Digital Forensics and Cybersecurity. *2021 9th International Symposium on Digital Forensics and Security (ISDFS)*. https://doi.org/https://doi.org/10.1109/isdfs52919.2021.9486354

The CAN-SPAM Act: Requirements for commercial emailers, Washington, D.C.: Federal Trade Commission, Bureau of Consumer Protection, Office of Consumer and Business Education. (2004). https://www.ftc.gov/business-guidance/resources/can-spam-act-compliance-guide-business

Vähäkainu, P., & Lehto, M. (2019). *Artificial intelligence in the cyber security environment*. https://www.researchgate.net/publication/338223306

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, L., & Polosukhin, I. (2017). *Attention Is All You Need*. http://arxiv.org/abs/1706.03762

Verma, R., Shashidhar, N., & Hossain, N. (2012). Detecting Phishing Emails the Natural Language Way. *Computer Security – ESORICS 2012*, 824–841. https://doi.org/https://doi.org/10.1007/978-3-642-33167-1_47

Wang, A., Singh, A., Michael, J., Hill, F., Levy, O., & Bowman, S. R. (2019). *GLUE: A Multi-Task Benchmark and Analysis Platform For Natural Language Understanding*. https://doi.org/https://doi.org/10.18653/v1/w18-5446

Weinstein, L. (2003). Spam wars. *Communications of the ACM*, *46*(8), 136. https://doi.org/https://doi.org/10.1145/859670.859703

Xu, K., Kliger, M., Chen, Y., Woolf, P. J., & Hero, A. O. (2009). Revealing Social Networks of Spammers Through Spectral Clustering. *2009 IEEE International Conference on Communications*. https://doi.org/https://doi.org/10.1109/icc.2009.5199418

APPENDICES

APPENDIX A

Setting Up the Environment (Mounting Google Drive for Google Colab)

```
#Mounting Google Drive
from google.colab import drive
drive.mount('/content/drive', force_remount=False)
```

```
import pandas as pd
#Reading CSV file as pandas Dataframe
df =pd.read_csv("/content/drive/MyDrive/spam.csv", header=
None, encoding="utf-8", error_bad_lines=False)
```

Dataset Cleaning and Reshaping Processes

```
#Removing na values from dataframe
def dt_na_value_cleaning(data):
  print("\nData Shape : ", data.shape)
  print("\nNull values before removal:: ")
  print(dt.isna().sum())

  dt.dropna(inplace=True)
  dt.reset_index(inplace=True,drop=True)

  print("\nNull values after removal: ")
  print(dt.isna().sum())
  print("\nData Shape after cleaning :" , dt.shape)

  return dt

# Removing duplicate values
def duplicate_content_removal(dt, col, ini_row):
    dt = dt.iloc[1: , :]
    print("\nNumber of data before removing duplicates: ",
 ini_row)
    duplicate_count = dt[col].duplicated().sum()
    print("\nNumber of Duplicates: ", duplicate_count)

    description_data = dt[col].drop_duplicates()
```

```
    cleaned_row = len(description_data)

    if (ini_row - cleaned_row) > 0:
        print("\nTotal data reduction : ", (ini_row - clea
ned_row))
        print("\Number of data after removing
duplicates is :", cleaned_row)
    else:
        print("\nNo duplicate data.")
    return list(description_data)

df=df.rename(columns={0: "E_Mail", 1: "Label", 2:"NaN"})
#deleting the unnecessary columns and rows
del df["NaN"]
df = df.iloc[1: , :]
df = df.replace(r'\n',' ', regex=True)
df = data_na_value_cleaning(df)
E_Mail = duplicate_content_removal(df, 'E_Mail', df.shape[
0])
df.shape
df['Label'].value_counts(normalize=True)
```

Splitting Dataset into Training and Test Dataset

```
# Randomize the dataset
data_randomized = df.sample(frac=1, random_state=1)

# Calculate index for split
training_test_index = round(len(data_randomized) * 0.8)

# Split into training and test sets
training_set = data_randomized[:training_test_index].reset
_index(drop=True)
test_set = data_randomized[training_test_index:].reset_ind
ex(drop=True)

print(training_set.shape)
print(test_set.shape)


training_set['Label'].value_counts(normalize=True)
test_set['Label'].value_counts(normalize=True)
```

Pre-processing

```
#Preprocessing text data

!pip3 install jpype1

from typing import List
from jpype import JClass, JString, getDefaultJVMPath, shut
downJVM, startJVM, java, isJVMStarted

#A function for Turkish letters, to fix upper case -
lower case issue
def trlower(metin):
    def trlower_(harf):
        if harf=='I': sonuc = 'ı'
        elif harf=='İ': sonuc = 'i'
        else: sonuc = harf.lower()
        return sonuc
    sonuc = ''
    for a in metin:
        sonuc += trlower_(a)
    return sonuc

#Function for removing digits
def sayi(metin):
    sonuc = ''.join([i for i in metin if not i.isdigit()])
    return sonuc
#Function for removing punctuation
def noktalama(metin):
    sonuc = "".join([i for i in metin if i not in string.p
unctuation])
    return sonuc

#Function for removing white-spaces
def wspace(metin):
    if metin is None:
        sonuc=''
    sonuc=metin.strip()
    return sonuc

#Function for removing words less than 3 characters
def remove_length(x):
    res = list()
    for word in x:
        if len(word) >= 3:
            res.append(word)
    return " ".join(res)
```

```
#lemmatization
ZEMBEREK_PATH = r'/content/drive/MyDrive/zemberek-
full.jar'

#startJVM function can be needed to be removed after one-
time run
startJVM(getDefaultJVMPath(), '-ea', -
Djava.class.path=%s' % (ZEMBEREK_PATH))

def lemmatizer(text):

    TurkishMorphology = JClass('zemberek.morphology.Turkis
hMorphology')
    morphology = TurkishMorphology.createWithDefaults()

    analysis: java.util.ArrayList = (
        morphology.analyzeAndDisambiguate(text).bestAnalys
is()
        )
    pos: List[str] = []
    for i, analysis in enumerate(analysis, start=1):
        f'\nAnalysis {i}: {analysis}',
        f'\nPrimary POS {i}: {analysis.getPos()}'
        f'\nPrimary POS (Short Form) {i}: {analysis.getPos
().shortForm}'

        pos.append(
            f'{str(analysis.getLemmas()[0])}'
            )
    return " ".join(pos)


#applying cleaning functions to training data set one at a
time to check the results
training_set["E_Mail"] = training_set["E_Mail"].apply(trlo
wer)
training_set["E_Mail"] = training_set["E_Mail"].apply(sayi
)
training_set["E_Mail"] = training_set["E_Mail"].apply(nokt
alama)
training_set["E_Mail"] = training_set["E_Mail"].apply(wspa
ce)
#Removing Turkish stopwords with NLTK library
training_set['E_Mail'] = training_set['E_Mail'].apply(lamb
da x: ' '.join([word for word in x.split() if word not in
(stop_words)]))
```

```
training_set['E_Mail'] = training_set['E_Mail'].str.split(
).apply(remove_length)


training_set["E_Mail"] = training_set["E_Mail"].apply(lemm
atizer)
#since ZEMBEREK lemmatizer gives UNK as output
if a word cannot be processed, replacing it with ‘’

training_set["E_Mail"] = training_set["E_Mail"].str.replac
e("UNK", '')
training_set.head()
```

APPENDIX B

Augmentation with BERT and distilBERT

```
!pip install nlpaug
!pip install transformers

import nlpaug.augmenter.char as nac
import nlpaug.augmenter.word as naw
import nlpaug.augmenter.sentence as nas
import nlpaug.flow as nafc

from nlpaug.util import Action


#BERTurk Augmentation Function
def augie1(text):
  aug = naw.ContextualWordEmbsAug(
      model_path='dbmdz/bert-base-turkish-
cased', action="substitute")
  augmented_text = aug.augment(text)
  return augmented_text

#distilBERTurk Augmentation Function
def augie2(text):
  aug = naw.ContextualWordEmbsAug(
      model_path='dbmdz/distilbert-base-turkish-
cased', action="substitute")
  augmented_text = aug.augment(text)
  return augmented_text

#applying BERT version of NLPAug
dfaug1=df
dfaug1["E_Mail"]= dfaug1.apply(lambda x: augie1(x["E_Mail"
]), axis=1)
#saving the augmented e-mails only
df12 =pd.read_csv("/content/drive/MyDrive/aug1.csv",  inde
x_col=0, encoding="utf-8", error_bad_lines=False)
df12.head()
#concatanating original emails with augmented emails with
BERT
concat1 = pd.concat([df, df12], axis=0)

#saving the augmented dataset as concat1 for BERT
with open('/content/drive/My Drive/concat1.csv', 'w', enco
ding = 'utf-8') as f:
  concat1.to_csv(f)
```

```
#applying distilBERT version of NLPAug
dfaug2=df
dfaug2["E_Mail"]= dfaug2.apply(lambda x: augie2(x["E_Mail"
]), axis=1)
dfaug2.head()
#saving the augmented e-mails only
with open('/content/drive/My Drive/aug2.csv', 'w', encodin
g = 'utf-8') as f:
  dfaug2.to_csv(f)

df22 =pd.read_csv("/content/drive/MyDrive/aug2.csv",  inde
x_col=0, encoding="utf-8", error_bad_lines=False)
df22

#concatanating original emails with augmented emails with
distilBERT
concat2 = pd.concat([df, df22], axis=0)


#saving the augmented dataset as concat2 for distilBERT
with open('/content/drive/My Drive/concat2.csv', 'w', enco
ding = 'utf-8') as f:
  concat2.to_csv(f)
```

APPENDIX C

Classification (First Approach)

```
import pandas as pd
#The CSV file here is the original dataset, for every
other case, the augmented datasets are used
df =pd.read_csv("/content/drive/MyDrive/spam.csv", header=
None, encoding="utf-8", error_bad_lines=False)

#Pre-processing and dataset shaping should be done as it
is shown in Appendix A


#creating the vocabulary
training_set['E_Mail'] = training_set['E_Mail'].str.split(
)
vocabulary = []
for email in training_set['E_Mail']:
  for word in email:
      vocabulary.append(word)

vocabulary = list(set(vocabulary))


len(vocabulary)#output: 6774


# Creating the dataframe to calculate probabilities on
word_counts_per_email = {unique_word: [0] * len(training_s
et['E_Mail']) for unique_word in vocabulary}

for index, email in enumerate(training_set['E_Mail']):
  for word in email:
    word_counts_per_email[word][index] += 1

word_counts = pd.DataFrame(word_counts_per_email)
word_counts.head()


training_set_clean = pd.concat([training_set, word_counts]
, axis=1)
training_set_clean.head()
```

```
#Isolating spam and ham messages first
spam_emails = training_set_clean[training_set_clean['Label
'] == 'spam']
ham_emails = training_set_clean[training_set_clean['Label'
] == 'ham']

# P(Spam) and P(Ham)
p_spam = len(spam_emails) / len(training_set_clean)
p_ham = len(ham_emails) / len(training_set_clean)

# N_Spam
n_words_per_spam_emails = spam_emails['E_Mail'].apply(len)
n_spam = n_words_per_spam_emails.sum()

# N_Ham
n_words_per_ham_emails = ham_emails['E_Mail'].apply(len)
n_ham = n_words_per_ham_emails.sum()

# N_Vocabulary
n_vocabulary = len(vocabulary)

# Laplace smoothing
alpha = 1


# Initiate parameters
parameters_spam = {unique_word:0 for unique_word in vocabu
lary}
parameters_ham = {unique_word:0 for unique_word in vocabul
ary}

# Calculate parameters
for word in vocabulary:
   n_word_given_spam = spam_emails[word].sum() # spam_mess
ages already defined
   p_word_given_spam = (n_word_given_spam + alpha) / (n_sp
am + alpha*n_vocabulary)
   parameters_spam[word] = p_word_given_spam

   n_word_given_ham = ham_emails[word].sum() # ham_message
s already defined
   p_word_given_ham = (n_word_given_ham + alpha) / (n_ham
+ alpha*n_vocabulary)
   parameters_ham[word] = p_word_given_ham
```

```
#Applying bayes theorem
def classify(email):

    email = re.sub('\W', ' ', email )
    email  = email.lower().split()

    p_spam_given_email = p_spam
    p_ham_given_email = p_ham

    for word in email:
        if word in parameters_spam:
            p_spam_given_email *= parameters_spam[word]

        if word in parameters_ham:
            p_ham_given_email *= parameters_ham[word]

    print('P(Spam|email):', p_spam_given_email)
    print('P(Ham|email):', p_ham_given_email)

    if p_ham_given_email > p_spam_given_email:
        print('Label: Ham')
    elif p_ham_given_email < p_spam_given_email:
        print('Label: Spam')
    else:
        print('Label: Ham')


def classify_test_set(email):

    email = re.sub('\W', ' ', email )
    email = email.lower().split()

    p_spam_given_message = p_spam
    p_ham_given_message = p_ham

    for word in email:
        if word in parameters_spam:
            p_spam_given_email*= parameters_spam[word]

        if word in parameters_ham:
            p_ham_given_ email *= parameters_ham[word]

    if p_ham_given_email > p_spam_given_email:
        return 'ham'
    elif p_spam_given_email > p_ham_given_email:
        return 'spam'
    else:
```

```
        return 'ham'

test_set['predicted'] = test_set['E_Mail'].apply(classify_
test_set)

#Accuracy Metrics

correct = 0
total = test_set.shape[0]

for row in test_set.iterrows():
    row = row[1]
    if row['Label'] == row['predicted']:
        correct += 1

print('Correct:', correct)
print('Incorrect:', total - correct)
print('Accuracy:', correct/total)


#False Positive and False Negative Metrics
fp=0
fn=0

for row in test_set.iterrows():
  row = row[1]
  if row['Label'] == 'spam':
    if row['predicted']=='ham':
      fn+=1
  if row['Label'] == 'ham':
    if row['predicted']=='spam':
      fp+=1
print('False Negative Ratio:', fn/total*100)
print('False Positive Ratio:', fp/total*100)
```

Classification (Second Approach)

```
!pip install pandas nltk
!pip3 install jpype1

import nltk
nltk.download('punkt')
nltk.download('stopwords')
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize
stop_words = set(stopwords.words('turkish'))
```

```
import warnings
warnings.filterwarnings('ignore')

import pandas as pd
import re
import string

from typing import List
from jpype import JClass, JString, getDefaultJVMPath, shut
downJVM, startJVM, java, isJVMStarted


%matplotlib inline
import matplotlib.pyplot as plt
import csv
import sklearn
import pickle
import numpy as np
from sklearn.feature_extraction.text import CountVectorize
r, TfidfTransformer
from sklearn.tree import DecisionTreeClassifier
from sklearn.model_selection import GridSearchCV,train_tes
t_split,StratifiedKFold,cross_val_score,learning_curve


#The CSV file here is the original dataset, for every
other case, the augmented datasets are used

df =pd.read_csv("/content/drive/MyDrive/spam.csv", header=
None, encoding="utf-8", error_bad_lines=False)

#Pre-processing and dataset shaping should be done as it
is shown in Appendix A

text = pd.DataFrame(df['E_Mail'])
label = pd.DataFrame(df['Label'])

#convert the text data into vectors
from sklearn.feature_extraction.text import TfidfVectorize
r

vectorizer = TfidfVectorizer()
vectors = vectorizer.fit_transform(df['E_Mail'])
vectors.shape


#features = word_vectors
features = vectors
```

```
#split the dataset into train and test set
X_train, X_test, y_train, y_test = train_test_split(featur
es, df['Label'], test_size=0.15, random_state=111

#import sklearn packages for building classifiers
from sklearn.linear_model import LogisticRegression
from sklearn.svm import SVC
from sklearn.naive_bayes import MultinomialNB
from sklearn.tree import DecisionTreeClassifier
from sklearn.neighbors import KNeighborsClassifier
from sklearn.ensemble import RandomForestClassifier
from sklearn.metrics import accuracy_score

#initialize multiple classification models
svc = SVC(kernel='sigmoid', gamma=1.0)
knc = KNeighborsClassifier(n_neighbors=49)
mnb = MultinomialNB(alpha=0.2)
dtc = DecisionTreeClassifier(min_samples_split=7, random_s
tate=111)
lrc = LogisticRegression(solver='liblinear', penalty='l1')
rfc = RandomForestClassifier(n_estimators=31, random_state
=111)

#create a dictionary of variables and models
clfs = {'SVC' : svc,'KN' : knc, 'NB': mnb, 'DT': dtc, 'LR'
: lrc, 'RF': rfc}

#fit the data onto the models
def train(clf, features, targets):
    clf.fit(features, targets)

def predict(clf, features):
    return (clf.predict(features))

pred_scores_word_vectors = []
for k,v in clfs.items():
    train(v, X_train, y_train)
    pred = predict(v, X_test)
    pred_scores_word_vectors.append((k, [accuracy_score(y_
test , pred)]))

#getting the accuracy scores of the classifiers using tf-
idf vectorizing
pred_scores_word_vectors
```

```
# Accuracy Score of Naïve Bayes
from sklearn.metrics import accuracy_score

y_pred_nb = mnb.predict(X_test)
y_true_nb = y_test

print(f"Accuracy of the classifier is: {accuracy_score(y_t
est, y_pred_nb)}")


#confusion matrix of Naive Bayes
from sklearn.metrics import confusion_matrix
from sklearn.metrics import plot_confusion_matrix

# confusion_matrix funnction a matrix containing the summa
ry of predictions
print(confusion_matrix(y_test, y_pred_nb))

# plot_confusion_matrix function is used to visualize the
confusion matrix
plot_confusion_matrix(mnb, X_test, y_test)
plt.show()

#Precision Score of Naïve Bayes
from sklearn.metrics import precision_score

print(f"Precision Score of the classifier is: {precision_s
core(y_test, y_pred_nb)}")

# Recall score of Naïve Bayes
from sklearn.metrics import recall_score

print(f"Recall Score of the classifier is: {recall_score(y
_test, y_pred_nb)}")

# F1-Score of Naïve Bayes
from sklearn.metrics import f1_score

print(f"F1 Score of the classifier is: {f1_score(y_test, y
_pred_nb)}"

# AUC-ROC Curve of Naïve Bayes
from sklearn.metrics import roc_curve, auc

class_probabilities = mnb.predict_proba(X_test)
preds = class_probabilities[:, 1]
```

```
fpr, tpr, threshold = roc_curve(y_test, preds)
roc_auc = auc(fpr, tpr)

# Printing AUC
print(f"AUC for our classifier is: {roc_auc}")

# Plotting the ROC
plt.title('Receiver Operating Characteristic')
plt.plot(fpr, tpr, 'b', label = 'AUC = %0.2f' % roc_auc)
plt.legend(loc = 'lower right')
plt.plot([0, 1], [0, 1],'r--')
plt.xlim([0, 1])
plt.ylim([0, 1])
plt.ylabel('True Positive Rate')
plt.xlabel('False Positive Rate')
plt.show()
```

Semantic Distance with BERT

```
!pip3 install torch torchvision torchaudio
!pip install sentence-transformers
from sentence_transformers import SentenceTransformer,util
import numpy as np

model = SentenceTransformer('emrecan/bert-base-turkish-
cased-mean-nli-stsb-tr')

def sem_difference(text1,text2):
  # encode sentences to get their embeddings
  embedding1 = model.encode(text1, convert_to_tensor=True)
  embedding2 = model.encode(text2, convert_to_tensor=True)
  # compute similarity scores of two embeddings
  cosine_scores = util.pytorch_cos_sim(embedding1, embeddi
ng2)
  return (100-(cosine_scores.item()*100))

dfnew=df
dfnew["New_E_Mail"]= df12["E_Mail"]
del dfnew["Label"]
dfnew['Difference'] = dfnew.apply(lambda x: sem_difference
(x['E_Mail'], x['New_E_Mail']), axis=1)


#Average of the distances
dfnew["Difference"].mean()
```